



2

NRL Report 9288

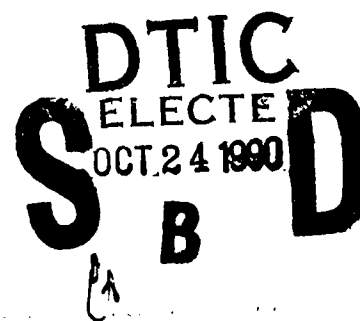
AD-A227 743

A Tandem Semantic Interpreter for Incremental Parse Selection

KENNETH WAUCHOPE

*Naval Center for Applied Research in Artificial Intelligence
Information Technology Division*

September 28, 1990



90 10 23 154

REPORT DOCUMENTATION PAGE			Form Approved OMB No 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE September 28, 1990		3. REPORT TYPE AND DATES COVERED
4. TITLE AND SUBTITLE A Tandem Semantic Interpreter for Incremental Parse Selection			5. FUNDING NUMBERS WU - 55-0230-0-0 TA - RS34-C74-000 PE - 62234N	
6. AUTHOR(S) Wauchope, Kenneth				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Research Laboratory Washington, DC 20375-5000			8. PERFORMING ORGANIZATION REPORT NUMBER NRL Report 9288	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Office of Naval Research Arlington, VA 22217			10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES (<i>software</i>)				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) <p>△ This report describes TINSEL (Tandem INTERpreter and SElection) a clause-local semantic interpreter and selection component for use with a natural language parser. As the parser incrementally regularizes candidate phrasal constituents into syntactic operator-operand forms, it submits each form to a user-defined selection process for the application of sublanguage constraints on co-occurrence patterns of semantic word classes. Only those analyses that pass selection are added to the parser's working space for involvement in further search, which can often reduce total processing time considerably. TINSEL treats selection as the enforcement of predicate domain constraints during semantic interpretation, the composing of a data structure that represents the meaning of a natural language phrase in some suitable formalism. The interpreter runs in tandem with the parser, transforming each semantically valid operator-operand form submitted to it into a predicate-argument form based on a set of declarative predicate models provided by the user. The parser's regularization component then composes partial interpretations of parent nodes from the interpretations of their children. TINSEL also operates top-down and so can be applied postparse to the final output of the parser rather than interleaved with it. (<i>128</i>)</p>				
14. SUBJECT TERMS Natural language understanding (NLU) Automated message analysis Semantic interpretation			15. NUMBER OF PAGES 50	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT SAR	

CONTENTS

INTRODUCTION	1
Selection	1
Selection in LSP	2
Implementing Selection in PROTEUS	2
Semantic Interpretation	3
Tandem Processing	4
Overview	5
Typographical Conventions	5
REGULARIZATION AND INTERPRETATION	6
PROTEUS Regularizations	6
Resolving Ambiguity	8
Normalizing Paraphrase	8
Selection and Sort Hierarchies	9
A Note on Case Grammar	10
What Selection Cannot Do	10
INTRODUCTION TO TINSEL	11
Predicate Models	12
Frame Definitions	12
Mapping Definitions	13
Top-Down Interpretation	14
Interpreting Quants	14
Interpreting Wffs	15
Bottom-Up Interpretation	16
Pruning the Search Space	17
Representing Global Ambiguity	18
Paraphrase Normalization	18
FORMAL OVERVIEW	19

Input Syntax	19
Output Syntax	19
Synopsis of TINSEL Macros	20
INTERPRETATION OF WFFS	21
Multiple Frame Mappings	21
Interaction with Strict Subcategorization	22
Depassivization	22
Regularization of Predicate Adjectives	23
Adjuncts	23
Wff SimpTerms	24
Conjoined Operands	26
INTERPRETATION OF SIMPTERMS	26
Underspecification	27
Constants	27
Quant-modifying Wffs	27
Quant-modifying RoleTerms	28
Argument	28
Adjunct	32
Implicit Relative Clause	32
NP Premodifiers	33
Inner Case Noun Premodifiers	34
Adjective Premodifiers	34
Other N-N Modifiers	35
Computing Denotations of NPs	35
ARGUMENT LABELING	36
The Case-Slot Identity Theory	37
Cases vs Thematic Relations	37
Beyond Synonymy and Inverses	38
COMPARISON TO OTHER APPROACHES	38
Linguistic String Parser	38
NP Conjoinings	39
Semantic Representations	39
Question-Answering System	40

Other Approaches	41
DISCUSSION	41
REFERENCES	43

Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	



A TANDEM SEMANTIC INTERPRETER FOR INCREMENTAL PARSE SELECTION

INTRODUCTION

The goal of natural language understanding (NLU) computer systems is to analyze and to use the information contained in English or other natural language discourse in much the same way as a human reader does. The natural language group at the Navy Center for Applied Research in Artificial Intelligence has been involved in building NLU systems for deriving extracts of Navy message texts [1,2] and as natural language interfaces to expert systems [3]. Our approach is based on techniques of computational linguistics (CL) and artificial intelligence (AI), treating language understanding as an activity based in both linguistic and world knowledge. Specifically, lexical knowledge recognizes words and their parts of speech, while syntactic knowledge discerns the structure of component phrases. Paralleling these are lexical and thematic semantics — knowledge of the meanings of individual words and the logical relationships among constituents. Finally, contextual knowledge about the discourse environment and the properties of the real world recovers implicit linguistic content, makes presuppositions, and draws inferences.

The syntactic analyzer we are currently using, the parser from the PROTEUS [4] system at New York University, applies lexical and syntactic knowledge (a dictionary and grammar) and generates a regularized parse of each sentence input to it. The piece of software we have developed to execute the next stage of analysis — lexical and thematic semantics — is called TINSEL* (Tandem INterpreter for SElection) and is the topic of this report. The module consists of about 1250 lines of Common LISP code and runs in Sun Common LISP on Sun-3/60, 3/260, and 4/60 workstations under UNIX. As the acronym suggests, TINSEL combines three key features — selection, semantic interpretation, and tandem processing — that distinguish it from, and improve on, approaches we have previously used.

Selection

In linguistics, a selection for a word *w* is that set of words with which *w* commonly co-occurs in a given syntactic relation [5]. In syntax-driven systems like ours, selections can be useful in pruning out bad structural analyses of sentences. This is particularly important when dealing with compact or "telegraphic" text, where the elision of syntactic cue words can vastly increase the number of grammatical ambiguities seen by the machine. An example from the Navy RAINFORM (tactical message) domain we have been investigating recently is the sentence *Crew aborted due to lack of comm* [unications], whose incorrect analysis "The crew was aborted..." can be ruled out by the application of a selection restriction (a process henceforth referred to simply as *selection*) requiring that the canonical object of the verb *abort* be a member of a set that includes such words as *search*, *operation*, and *attack* (*The search was aborted*). When working within a constrained subset of English (called a sub-language) like the language of Navy tactical messages, these selection sets tend to exhibit regularities of meaning that permit them to be regarded as semantic word classes. Hence, in the RAINFORM domain,

Manuscript approved July 2, 1990.

*tinsel (tin'sl), *n., adj.* 1. a glittering, scintillating material decorating a tree. 2. any showy pretense; gaudy; tawdry. Naturally it is the former definition that applies here.

the set {*search*, *operation*, *attack*...} corresponds to a category called **stask** of words connoting ship tasks.

Although the PROTEUS parser is a direct descendant of the Linguistic String Project (LSP) parser [6] with which our initial Navy message understanding work was done, it was not possible for us to simply adapt LSP's treatment of selection and semantic representation directly to PROTEUS. To understand this, we need to briefly compare the philosophies of the two systems, first regarding selection and then semantic representation.

Selection in LSP

The LSP system consists of a syntactic analyzer, regularization component, and information formatting component. The syntactic analyzer is a parser for context-free grammars (CFGs) that have been augmented with procedural tests called restrictions (written in a high-level language called Restriction Language) that subject each candidate phrase structure analysis to well-formedness tests and selectional constraints before acceptance. After parsing and selection, the LSP regularization component restructures the parse tree thus obtained into a more normalized form by using transformation operations. Finally, the information formatting component maps each regularized tree into a tabular or slot-filler structure called an information format.

The selectional constraints in LSP are encoded as syntactic co-occurrence patterns over the semantic categories of the domain sublanguage. During parsing, grammar restrictions apply these patterns to the semantic classes of the lexical items in the (unregularized) parse tree. For each semantic category to which a lexical item belongs, the lexicon writer has assigned the category as a feature (lexical attribute) of the item. In our example, the words *crew*, *abort*, and *search* might be assigned the lexical semantic categories **org**[anization], **status**, and **stask**, respectively. During parsing of *Crew aborted*, the verb-subject-object selection pattern

```
LIST V-S-O = STATUS: (ORG: (STASK, NULLOBJ))
```

accepts the analysis "The crew aborted [something]" by permitting the subject of a **status** verb to be in class **org**, but rejects the incorrect analysis "The crew was aborted" by requiring that the object (if not empty) be in class **stask**.

LSP is a batch processor written in FORTRAN, and so is not as flexible as we would like for use in AI work, where an interactive, interpretive software environment is desirable. It is also a backtracking parser, which is slower and less efficient on ambiguous input than other designs. For those reasons, we recently acquired the parser from LSP's direct descendant PROTEUS, an interactive chart parsing system written in Common LISP, and have begun using it as our main syntactic analyzer.

Implementing Selection in PROTEUS

PROTEUS is also a parser for restriction-augmented CFGs, but it has adopted a different philosophy regarding regularization and selection than its predecessor. PROTEUS accepts grammars in which each production specifies a pattern by which the phrase's regularized form is to be constructed. For efficiency, the parser only computes regularized forms at particular grammatical nodes that the user has specified. Whenever it constructs one of these nodes, the parser invokes a user-defined selection function before adding the node to its working memory. The intention is that selection be applied to the regularized form that has just been composed, rather than (as in LSP) to the phrase structure tree itself. To illustrate, the PROTEUS regularization for *Crew aborted attack* might be

```
(PAST ABORT (NULL-DET Y1 CREW SINGULAR)
 (NULL-DET Y2 ATTACK SINGULAR)),
```

which identifies ABORT as the main verb or predicate, CREW as its first (subject) operand, and ATTACK as its second (object) operand, a form suitable for application of the selection restrictions discussed earlier.

In LSP, the semantic classes of the words in the sentence are stored in the parse tree as node attributes, so the parser can execute selection by grammar restrictions operating on the tree itself. PROTEUS Restriction Language, however, does not provide operators for examination of the regularized form, which is composed from a special lexical feature called the translation attribute that is intended to represent lexemic rather than semantic content. Instead, the PROTEUS philosophy is that semantic classes and selection patterns be realized as LISP data structures and processes independent of the grammar and parsing mechanism. This has the advantage of providing a clean separation of syntax and semantics into distinct modules, which nonetheless can still work in tandem.

With this philosophy in mind, TINSEL was designed to be independent of any particular syntactic grammar or parsing methodology. It is based on a formal specification of the regularized forms that constitute its input, and so can be used with any parser, grammar, and lexicon that generate syntactic regularizations of the same form as PROTEUS translations. The TINSEL user defines semantic classes and selection patterns (called predicate models) for each domain in simple declarative statements that can either be invoked independently of the PROTEUS lexicon macros, or be included in the same data files as the lexicon if desired. The semantic classes can be organized into type hierarchies, allowing selection patterns to be as broad or as precise as desired.

Semantic Interpretation

Although selection is useful in isolating the correct parse of a sentence, the end goal of natural language analysis is not a phrase structure tree, but a representation of the utterance's meaning. Semantic interpretation is the composing of a data structure that represents the meaning of a natural language phrase in some suitable formalism, typically logic-based. In our work with LSP, the information format table served as a specialized kind of partial semantic representation. The table contains a row for each simple sentence in the text and a column (slot) for each basic class of data that can occur in a sentence of a particular informational type. The entries in the table are individual words or strings of words from the input sentence, and the slot headings represent keys for use in data retrieval.

Since information formats were designed specifically for text retrieval applications, they have been used successfully as a textual database schema of sorts, but they are not suitable as a general-purpose semantic representation for AI applications, as has been discussed in an earlier report [2]. In line with current trends in CL- and AI-based natural language processing, we wanted our new PROTEUS-based system to generate an application-neutral meaning representation composed of tokens representing concepts and semantic relationships, not a *text* representation composed of English words and information-retrieval keys. In this regard, TINSEL "kills two birds with one stone" by treating selection as an aspect of semantic interpretation, not simply as the enforcement of word co-occurrence patterns. Selectional constraints in TINSEL are applied during the transformation of the PROTEUS regularization into a first-stage semantic structure (sometimes called *logical form*) suitable for further knowledge-based processing.

To illustrate, a skeletal logical form for the sentence *Crew aborted attack* might be

```
(PAST discontinue :isa action
      :agent (ship-personnel :isa org)
      :patient (hostile-act :isa stask)).
```

During construction of this form from the PROTEUS regularization seen earlier, TINSEL applies the selectional constraints associated with the **discontinue** predicate to ensure that the :agent (the

intentional doer of the action) is in semantic class **org**, and the :patient (the entity whose state is affected by the action) is in semantic class **stask**. The :isa relation establishes the class-subclass relationships that make these selections possible: for example, **ship-personnel** passes the selectional constraint on the :agent slot by being a subclass of **org**.

Logical form has several advantages over the implementation of information formats used in our prior work. It regularizes the words of the sentence into their underlying conceptual identities (e.g., *abort* → **discontinue**), so that a paraphrase like *Crew abandoned assault* can receive the same interpretation as *Crew aborted attack*. Logical form also distinguishes the predicate of the expression (**discontinue**) from its arguments (**ship-personnel** and **hostile-act**), and assigns generalized argument types like :agent and :patient (grounded in a linguistic theory called case grammar [7]) that make explicit the arguments' relationships to the predicate. For example, the :patient argument explicitly represents that it was the attack that was discontinued, not the crew. Finally, logical form allows host-modifier relations to be nested arbitrarily deeply rather than forced into either flat or connected sentential forms, thus preserving the logical relationships inherent in the original utterance.

Tandem Processing

Selection can be applied either during or after parsing, but the former permits the active pruning of semantically anomalous nodes from the parser's search space, thus minimizing the amount of

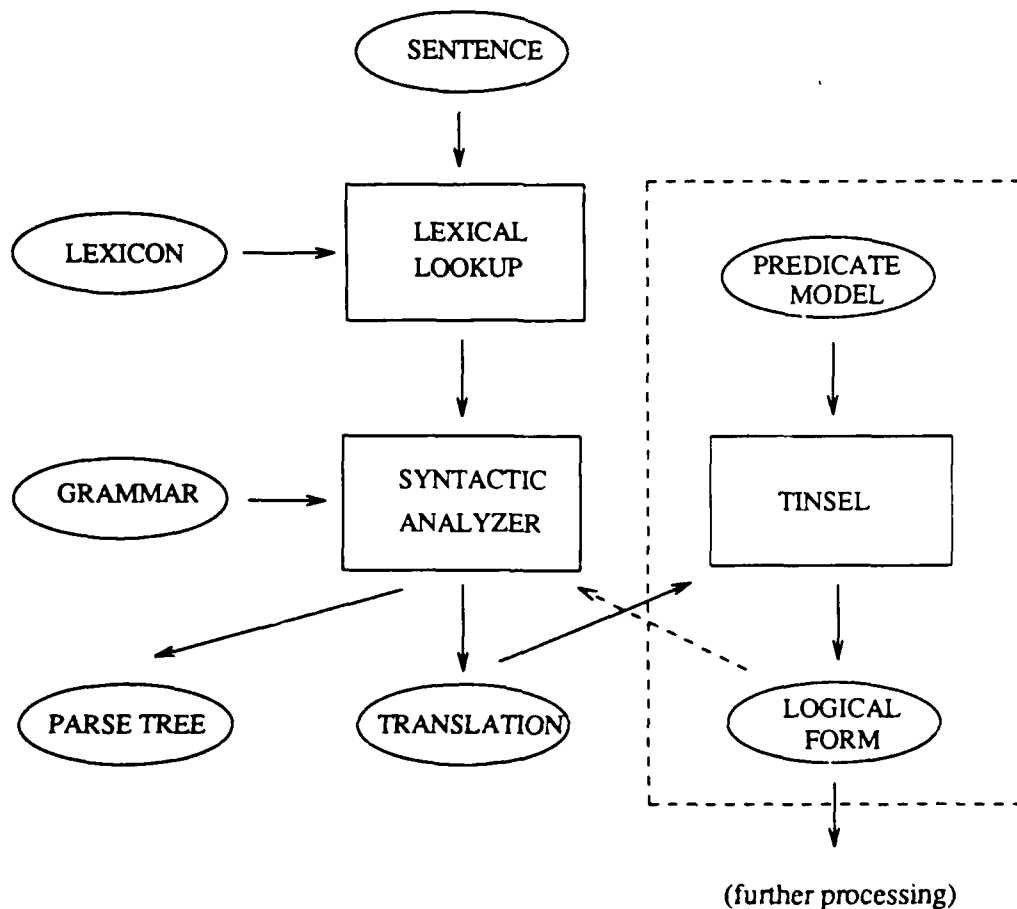


Fig. 1 — NLU system architecture

processor time spent pursuing bad parses. When parsing the sentence *Crew aborted due to lack of comm*, for instance, rejection by selection of the bad subparse "The crew was aborted" means that the parser does not have to pursue the more advanced analysis "The crew was aborted due to lack of communications," but can abandon that particular path immediately.

When run in tandem mode (interleaved with the parser), TINSEL uses the parser's regularization component to incrementally compose partial semantic interpretations of component phrases of the input. It notifies the parser of whether an interpretation was achieved for each subparse, allowing the parser to either add that node to its search space or discard it. Hence, the pruning of the parser's search occurs as a "side effect" of semantic structure building, which can be considered TINSEL's primary function. Finally, since sequential processing can also be desirable in certain circumstances (as when debugging grammars and semantic models), TINSEL can be applied postparse as well. Figure 1 shows the architecture of our NLU system, with the interpreter (dashed enclosure) transforming PROTEUS regularizations into logical forms. The dashed arrow represents the feedback from the interpreter to the parser when the two are run in tandem. When interpretation is complete, TINSEL's output is then passed on to other (projected) modules for further processing into a final meaning representation suitable for such potential Navy applications as database query/update, message classification and sorting, and text summarization.

Overview

This report begins with a description of PROTEUS syntactic regularizations and how selection is executed during the transformation of these forms into first-stage semantic interpretations. This is followed by an introduction to the TINSEL interpreter and the declarative predicate models by which the user specifies selectional constraints and interpretations. The report then explores in more detail various issues in semantic interpretation that TINSEL addresses. This is followed by a discussion of case grammar and related theories relevant to semantic interpretation, and a comparison of our interpreter and its output to several other computational approaches that have appeared in the literature. The report concludes with a discussion of some issues in semantic interpretation that TINSEL does not yet fully address.

Typographical Conventions

The report uses the typographical conventions shown in Table 1. "Typewriter" fonts are used to represent the data that are processed and generated by PROTEUS and TINSEL. *Italics* denote the lexical data (sentences, phrases, and words) that are the parser's input. UPPERCASE type is used for the regularized forms that are output by the parser and input to the semantic interpreter; the regularizations of phrases and subphrases are enclosed in parentheses. In the interpreter's output (logical form), which is also parenthesized, **bold** typeface denotes a semantic class, and a :colonized symbol is a semantic role. Note that token names are free to overlap among these four data types; for instance, the English word *submarine* could translate to the regularization token SUBMARINE just as the word *sub* does, and SUBMARINE in turn maps (in our example) to the semantic token **submarine**. Similarly, the word *for* has been regularized as the token FOR, which the user could map to a semantic role

Table 1 — Illustration of Typographical Conventions

Data Type	Example
word/phrase	<i>for sub</i>
regularized word	FOR, SUBMARINE
regularized phrase	(FOR (NULL-DET Y2 SUBMARINE SINGULAR))
logical form	(:goal (NULL-DET Y2 (:class submarine) SINGULAR))
semantic class	submarine
semantic role	:goal, :class
LISP function	get-semantics, defframe, defnpred

named :for (rather than :goal) if so desired. Finally, lowercase type denotes LISP functions and macros available to the TINSEL user.

REGULARIZATION AND INTERPRETATION

The output of PROTEUS is a regularized syntactic form (also called a translation or operator-operand form) that has a logiclike structure but has not yet undergone semantic interpretation. In this section, we introduce the issues involved in the development of a lexical/thematic interpreter for PROTEUS translations.

PROTEUS Regularizations

PROTEUS regularizations are generated from two sources. First, each word in the lexicon has a "translation" attribute that specifies the word's regularized form. The default is the morphological root (including tense or number information if appropriate), but the user can override this by specifying an explicit alternative. Second, each production in the grammar contains a pattern called a Translation Rule that specifies how the node's regularization is to be composed from those of its children.* The resulting structures should conform to a logic-like representation language called Simplified Operator/OPerand language, or SOOP [8]. Figure 2 shows the syntax of the SOOP language.

As an illustration, PROTEUS regularizes the sentence *Ctc was passed to relief as the Wff (well-formed formula)*

```
(PAST PASS ANYONE
 (NULL-DET Y1 CONTACT SINGULAR)
 (TO (NULL-DET Y2 RELIEF SINGULAR))).
```

Figure 3 shows the matchup between this Wff and the SOOP syntax. PAST is an Op1 (unary operator) denoting the Wff's tense, and PASS is the Pred (predicate) of the Wff. The next three forms (the operands of the Wff) are Terms. The first two are SimpTerms (simple terms), a Con (constant) and Quant (quantified expression), respectively. Note that PROTEUS has depassivized the sentence by providing the constant ANYONE as the first operand of the Wff, and that it regularizes the abbreviation *ctc* into its underlying root form CONTACT. The components of the Quant are the Q (quantifier) NULL-DET, Var (identifier variable) Y1, Npred (noun predicate) CONTACT, and Number SINGULAR. The third operand of the Wff is a RoleTerm, consisting of the Role TO followed by a second Quant. The first two operands of the Wff can be considered to have the implicit positional roles of subject and object, respectively. Table 2 shows the most common syntactic structures from which each of these SOOP constructs is

<Wff>	::= "(" <*Op1> " <*Pred> <Term> ")".
<Wff>	::= "(" <*Op2> <Wff> <Wff> ")".
<Wff>	::= "(lambda (" <*Var> ") " <Wff> ")".
<Term>	::= <SimpTerm> <RoleTerm>.
<SimpTerm>	::= <*Con> <Quant>.
<RoleTerm>	::= "(" <*Role> <SimpTerm> ")".
<Quant>	::= "(" <*Q> <*Var> <*Npred> <*Number> <Mod> ")".
<Mod>	::= <Amod> <RoleTerm> <Wff>.
<Amod>	::= <*Apred> "(" <*Role> <*Att> ")".

Fig. 2 — SOOP syntax

*See [8] for a discussion of Translation Rule Language and the relationship between PROTEUS regularizations and grammatical phrase structure rules.

<Wff>	<*Op1>	PAST				
	<*Pred>	PASS				
	<Term>	<SimpTerm>	<*Con>	ANYONE		
	<Term>	<SimpTerm>	<Quant>	<*Q>	NULL-DET	
				<*Var>	Y1	
				<*Npred>	CONTACT	
				<*Number>	SINGULAR	
	<Term>	<RoleTerm>	<*Role>	TO		
			<SimpTerm>	<Quant>	<*Q>	NULL-DET
					<*Var>	Y2
					<*Npred>	RELIEF
					<*Number>	SINGULAR

Fig. 3 — SOOP structure of example Wff

Table 2 — SOOP Type Correspondences

SOOP Type	Syntactic Type
Wff	clause
Op1	tense, modal
Op2	coordinating conjunction
Pred	verb, predicate adjective
Con	pronoun, proper name
Quant	noun phrase
Q	determiner
Npred	noun
Apred	predicative adjective
Att	attributive adjective
RoleTerm	prepositional phrase, subordinate clause
Role	preposition, subordinating conjunction

derived. Since this report will make frequent reference to "Wff," "Pred," "Quant," and the other SOOP constructs, the reader is urged to refer to Figs. 2 and 3 and Table 2 for clarification of these terms whenever necessary.

As a logiclike language, SOOP closely resembles a predicate-argument notation such as the logical form Allen [9] uses for his first-stage semantic interpretations. Allen defines logical form as a level of representation intermediate between syntactic structure and logic, derived using information strictly local to the sentence. The motivation behind producing both an intermediate logical form and final meaning representation is that fundamentally different kinds of knowledge are involved in generating the two: the logical form can be built on a strictly sentence-local (lexical/syntactic) basis, whereas the mapping to the final representation is dependent on contextual knowledge (intersentential information and world knowledge). The derivation of logical form consists largely of word sense disambiguation, which is one natural byproduct of the selection process.

PROTEUS translations, however, are strictly syntactic regularizations suitable for (but not yet having undergone) selection. They can thus be viewed as a level of representation intermediate between parse tree and logical form, resembling the latter in structure but still consisting only of lexemic and syntactic information. The close resemblance of the PROTEUS regularization to Allen's logical form suggests that a predicate-argument representation would be ideal for PROTEUS semantic interpretations. Two main operations are necessary to transform SOOP expressions into true predicate-argument forms:

1. Map from Preds and Npreds to semantic classes (i.e., typed n-ary predicates);
2. Map from Roles (including implicit positional roles like subject and object) to predicate argument positions.

If these mappings were simply one-to-one, they would only constitute a trivial renaming, but the mappings can be both one-to-many and many-to-one. One-to-many mapping resolves lexical and thematic ambiguity, and many-to-one mapping results in the semantic normalization of paraphrase, as follows.

Resolving Ambiguity

Lexical ambiguity can be divided into categorical ambiguity, polysemy, and homonymy. Words that are categorically ambiguous have meanings that vary with the part of speech, as in *Ross saw the sink sink*. A polysemous word has several meanings (within a single part of speech) related to each other, as in *Nadia opened the door* and *Nadia opened a new business*. In homonymy, the meanings are unrelated, as in *the dog's bark* and *the tree's bark*. To deal with categorical ambiguity, a semantic interpreter for SOOP must provide separate predicate mappings for Preds and Npreds, such as mapping the Npred SINK to the sort predicate **basin** and the Pred SINK to the nonsort predicate **submerge**.^{*} Polysemy and homonymy simply require that each Pred (e.g., OPEN) or Npred (BARK) be allowed to map to more than one predicate.

Thematic ambiguity is ambiguity in the mapping from syntactic role to semantic role. An example from Fillmore [10] is the sentence *I copied that letter*, which can be uttered when pointing either to the original letter or to the copy. The sentence is not structurally ambiguous, nor are the words *copy* or *letter* polysemous, so the ambiguity must arise from the inadequacy of syntactic structural roles (here, object) to capture underlying logical relationships. Similarly, the syntactic construct PP (prepositional phrase) can be ambiguous regarding the semantic relationship it connotes: *Ross went to the basement for Nadia* can either mean he went there to get her, or that he went there on her behalf.

To handle thematic ambiguity, the mapping from Pred or Npred to predicate must also include a mapping from each Term the Pred or Npred can host to the appropriate argument position of the predicate. Assuming that the predicate **make-duplicate** takes three arguments we might call :agent, :source, and :destination, the Pred COPY should map the subject to the :agent argument of **make-duplicate**, and map the object to either :source or :destination, which accounts for the ambiguity of *I copied that letter*—either *that letter* (the sentential object) is the :source from which a copy has been made, or is the copy itself (:destination). Finally, COPY should map a FROM operand strictly to the :source argument, generating a single unambiguous interpretation for *I copied that letter from this one* since no argument — here, :source — can be filled more than once.

Normalizing Paraphrase

A semantic interpreter's second function is the many-to-one mapping of different regularizations to the same predicate-argument structure. Just as one-to-many mapping explicates lexical and thematic ambiguity, normalization resolves lexical and thematic paraphrase, or equivalence of meanings. An interpreter handles lexical paraphrase (synonymy) by allowing more than one Pred/Npred to map to the same predicate, and handles thematic paraphrase by allowing more than one syntactic role marker to map to the same argument. Hence, *Ross gave the scouts money* and *Ross donated money to*

^{*}In logic, a sort predicate ascribes class membership to a variable; a nonsort predicate ascribes properties to or relationships among its (one or more) arguments.

the scouts can both be normalized to the predicate form

transfer-ownership (:agent **ross**, :theme **money**, :to-poss **scouts**)

by mapping both Preds GIVE and DONATE to the predicate **transfer-ownership**, and mapping both indirect object and TO to the predicate argument :to-poss.*

Selection and Sort Hierarchies

If simply mapping from syntactic role to predicate argument were sufficient to form an interpretation, *Ross went to the basement for his fiddle* could be interpreted as his going there on the instrument's behalf. To be interpreted as the :beneficiary argument, however, FOR's complement must be in the selection set corresponding to the **person** sort predicate. In both ambiguity resolution and synonymy normalization, selection determines whether a mapping from a Pred/Npred and its operands to a particular predicate and its arguments can succeed. For our purposes, then, a selection for a word is more properly defined as the set of words that commonly co-occur in a given thematic (not syntactic) relation to the word. *For his fiddle* and *for Nadia* both fill the same syntactic relation to the verb *go*, but can fill different thematic relations. Therefore, syntactic relations should be mapped to thematic relations before selection restrictions are applied.

In this light, selection can be treated as the enforcement of predicate domain constraints during semantic interpretation [11]. To continue our example of *Ross went to the basement for his fiddle*, we could model GO ('he root form of the verb *went*) as the four-argument predicate **deliberate-go**:

```
deliberate-go
  :agent person
  :to-loc location
  :beneficiary person
  :goal thing

GO → deliberate-go:
  subject → :agent
  TO → :to-loc
  FOR → :beneficiary, :goal
```

The domain of (that is, the selectional constraint on) the :beneficiary argument is the semantic class **person**, and the domain of the :goal argument is **thing**. Although FOR maps to both :beneficiary and :goal, *for his fiddle* cannot be interpreted as the former since *fiddle* is not a member of the selection set corresponding to **person**.

We want the word *fiddle* to have the sort predicate **violin** as its lexical interpretation, however, not **thing**. Thus, for *Ross went for his fiddle* to receive an interpretation, a sort hierarchy is needed to establish that all **violins** are **things**. A well-structured sort hierarchy allows newly added predicates to automatically assume the proper sorts for selection. For example, if we needed a new class **musical-instrument** to serve as (one of) the lexical interpretations of the noun *instrument* and as a selectional constraint on the verb *tune*, we could interpose **musical-instrument** between **thing** and **violin** in the hierarchy (all violins are musical instruments and all musical instruments are things). Then the sentences *Ross went for his instrument* and *Nadia tuned her fiddle* would automatically pass selection.

*The question of whether *The scouts received money from Ross* (where RECEIVE is an inverse of GIVE rather than a synonym) can also be normalized to this predicate is addressed in the section "Argument Labeling."

A Note on Case Grammar

Thematic argument names like :agent and :beneficiary derive from Fillmore's [7] linguistic theory called case grammar. The main points of Fillmore's theory are:

- There is a small, closed set of basic verbal argument types that he calls "deep structure cases."
- General mapping rules determine how each case is syntactically marked, either positionally (subject, object, or indirect object) or prepositionally. For example, if a sentence contains an :agent, any :instrument must be marked with (*Ross broke the window with a hammer*), otherwise the :instrument must be marked (positionally) as subject (*The hammer broke the window*).
- Each case can be filled at most once in each sentence.
- Each verb has an associated "case frame" that spells out which cases the verb has and how they can co-occur.

The appeal of this theory in natural language processing is that it forms a bridge between a purely syntactic (subject-verb-complement) analysis of language and a semantic (predicate-argument) one. Cases like :agent and :beneficiary are syntactico-semantic roles in much the same way that selection sets represent syntactico-semantic word categories. Since the cases have direct syntactic correlates, case grammar also fits naturally into the sublanguage approach: verbs with similar case frames (i.e., similar complement distributions) tend to have shared or equivalent meaning, thus forming semantic patterns.

Case grammar provides a discipline that can be useful in clarifying whether a particular operand represents an argument or an adjunct, and in deciding whether two verbs are truly synonymous. For example, the verbs *give* and *send* have similar syntactic behavior and selection sets (*Nadia gave Ross a marmot*, *Nadia sent Ross a marmot*), so ordinary subject-verb-object distributional analysis would regard them as semantically equivalent. But since the first takes a :to-poss (final possessor) argument and the second :to-loc (final location), the two verbs cannot be mapped to the same case frame representation. This is appropriate since Ross can be inferred to possess the marmot in the first sentence, but he may not have even received it in the second.

Different versions of case grammar postulate widely different sets of roles, but since TINSEL does not require adherence to any particular set of argument names (nor does it enforce generalized case-marking rules, for reasons to be explored later), no attempt at presenting an exhaustive or well-defined set is made in this report. To illustrate one possible set of thematic roles, however, Table 3 lists the relations (many adapted from Ref. 9) used in the examples in this report, along with synopses of their semantics and illustrations of each.

What Selection Cannot Do

As only the first phase of enforcement of what Jackendoff [12] calls "well-formedness conditions on semantic interpretations," selection's role is to filter out those analyses that are so semantically ill-formed as to be uninterpretable, not those whose interpretations are false. For example, the sentence *We detected the aircraft with sonar* is not meaningless, since aircraft can be detected and things can be detected with sonar. It is just false, since sonar happens to operate in water rather than air. Such inter-operand consistency checking should be handled by a later stage of pragmatic (world-knowledge) analysis rather than by selection semantics.

Table 3 — Example Set of Thematic Relations

Relation	Synopsis	Example
:agent	intentional doer	Nadia baked the brownies
:experiencer	mental undergoer	Ross loves brownies
:patient	affected entity	The oven exploded
:theme	unaffected verb topic	Ross knows the answer
:instrument	inanimate means	The hammer drove the nail
:destination	thing produced	Nadia knitted a sweater
:source	material of origin	Nadia knitted it from wool
:at-poss	possessor	Ross owns a car
:from-poss	original possessor	Ross bought the car from Nadia
:to-poss	final possessor	Nadia sold Ross the car
:at-loc	location	The box contained a marmot
:from-loc	original location	Nadia took it out of the box
:to-loc	final location	Nadia put the marmot on the shelf
:beneficiary	beneficiary	Nadia baked Ross the brownies
:goal	objective	Nadia wants to be a star

Since selection as defined here operates on types (sort predicates), it is constrained by what can constitute a reasonable type. Allen notes that useful types tend to correspond to individual English words rather than relational phrases. So if one were tempted to enforce a selectional constraint against *#a spoonful of elephants** by having *spoonful* select only for arguments that are **smaller-than-a-spoon** (a "class" to which *elephant* would not belong), Allen's criterion would reject that class as representing an *ad hoc* type. It is more reasonable to model the selection set for *spoonful* as simply **material**, and later apply contextual world knowledge of relative sizes to judge the phrase as unreasonable.

It is also not within the realm of predicate-local interpretation to analyze an NP (noun phrase) whose denotation is different from that of its head noun. For an interpreter to reject *#The melted ice shattered*, it would have to recognize that the entity denoted by *melted ice* has class attribute **fluid** even though the NP's head noun *ice* has been assigned **solid** in its lexical semantics (so that *the ice shattered* will pass selection). Our approach regards **melt** as a predicate with domain **solid**, not a function with range **fluid**: *Melted ice* is ice that it is true has melted, not that which results from applying the function **melt** to ice. The knowledge that *melted ice* is not a solid (whereas *melting ice* is, for example) again falls under the scope of world knowledge, not selection.

Finally, predicate-argument selection can only operate on forms that contain explicit or easily recoverable predicates, making it an inappropriate technique for dealing with complex, ambiguous NPs. For example, *rubber baby buggy bumper* requires considerable world knowledge to be correctly understood as a "bumper made of rubber that is part of a buggy that carries a baby." Such referring expressions are typically handled by a separate NP-dereferencing component that can take both world knowledge and discourse context into account.

INTRODUCTION TO TINSEL

We begin with an overview of the basic principles of TINSEL. Later sections take a more in-depth look at the details of its behavior.

*The stigma # denotes a semantically unacceptable phrase.

Predicate Models

TINSEL is at heart an interpreter, a data-driven program that analyzes statements in one declarative language (SOOP expressions) using patterns in another declarative language, the predicate models. TINSEL predicate models are composed of two basic parts, frame definitions and mapping definitions.

Frame Definitions

The user specifies each predicate in the sublanguage domain using the TINSEL `defframe` (define frame) macro, whose syntax is shown in Fig. 4. Figure 5 illustrates some sample frame definitions for the RAINFORM message domain. Frame classes and slots can be given whatever names the user chooses (except for the distinguished slot `:isa`, used to establish the sort hierarchy). In our example, the root of the hierarchy is a predicate class we've called **all**. This has been given subclasses **thing**, **state**, and **process**, which are roughly the semantic equivalents of common nouns (0-ary or sort predicates), adjectives (unary attributive predicates), and verbs (n-ary relation/event predicates), respectively. Multiple inheritance is supported, as seen in the definition of **craft**. Terminal classes like **p-3c** and **transfer-data** refer to the most specific types of entities in the domain, here a class of reconnaissance/patrol aircraft and a data transfer action, respectively.

The frame name(s) following each slot represent the selectional constraint on the contents of the slot. For example, since **craft** is both an **equip** and an **org**, **p-3c** can pass selection both as the `:patient` of **equip-fail** (*The P3 malfunctioned*) and as the `:agent` of **transfer-**

<code><Defframe></code>	<code>::= "(defframe" <*FrameClass> [" :isa" <Constraint>] <SlotSpec>* ")".</code>
<code><SlotSpec></code>	<code>::= <*SlotName> <Constraint>.</code>
<code><Constraint></code>	<code>::= <*FrameClass> "(" <*FrameClass>+ ")".</code>

Fig. 4 — `defframe` macro syntax

(defframe all)
(defframe thing :isa all)
(defframe state :isa all)
(defframe process :isa all)
(defframe physobj :isa thing)
(defframe equip :isa physobj)
(defframe org :isa thing)
(defframe asw-commander :isa org)
(defframe craft :isa (equip org))
(defframe aircraft :isa craft)
(defframe p-3c :isa aircraft)
(defframe detected-craft :isa craft)
(defframe data :isa thing :theme process :source equip)
(defframe detection-data :isa data)
(defframe equip-fail :isa process :patient equip)
(defframe depart :isa process :theme craft :from-loc location :to-loc location)
(defframe detect :isa process :agent org :theme org)
(defframe mutual-contact :isa process :agent org :theme org)
(defframe go-past :isa process :agent craft :at-loc physobj)
(defframe transfer-data :isa process :agent org :theme data :to-poss org)

Fig. 5 — Sample frame definitions for RAINFORMs

data (*The P3 relayed the information*).

The **defframe** macro also supports slot/constraint inheritance. Thus, the frame **transfer-data** could instead have been implemented as illustrated in Fig. 6. Under this approach, **transfer-data** inherits the **:agent** slot (and the slot's constraint **org**) from its grandparent **action**, and the **:to-poss** slot and constraint from its parent **transfer**. It also inherits the **:theme** slot from **action**, but respecifies its copy of the slot to tighten its constraint from the overgeneral **thing** to the more specific **data**. Whenever a new agentive action is added to the model it can be made a subclass of **action**, whereby it will automatically inherit an **:agent** slot with selectional constraint **org**. Organizing nonsort predicates into a hierarchy can also prove useful in selection; for example, the **action** class could be used to constrain the infinitival complement of a verb like *order* to be an agentive action (*Nadia ordered Ross to whistle*, #*Nadia ordered Ross to know the answer*).

```
(defframe process :isa all)
(defframe action :isa process :agent org :theme thing)
(defframe transfer :isa action :to-poss org)
(defframe transfer-data :isa transfer :theme data)
```

Fig. 6 — Illustration of slot/constraint inheritance

Mapping Definitions

The user must also map each Pred and Npred in the lexicon to the appropriate frame class(es), using the TINSEL macros **defpred** (define Pred) and **defnpred** (define Npred), respectively. Figure 7 shows the syntax of these forms. Figure 8 shows some sample mapping definitions for the Preds **DEPART**, **PASS**, and **RELAY** and the Npreds **P3**, **INTELLIGENCE**, **INFORMATION**, **AX**, **ASWC**, **CONTACT**, and **DEPARTURE**.

Each invocation of **defpred** or **defnpred** begins with the name of the Pred/Npred being defined, followed by one or more frame mappings. Each frame mapping names a predicate frame the token can map to, followed by zero or more slot mappings specifying how each of the frame's slots (if any) can be filled by an operand of the PROTEUS regularization. For example, either an **O** (object) or **FROM** operand of a **DEPART** Wff can map to the **:from-loc** argument of the predicate **depart**, allowing that predicate to match both *Lsft* departed the area* and *Lsft departed from the area*. **S**, **IO**, and **O** are TINSEL's mnemonics for the implicit positional roles of unmarked (SimpTerm) operands: "subject" (operand 1), "indirect object" (operand 2 if there is a direct object), and "object" (operand 2 if there is no indirect object, otherwise operand 3). For example, the regularization of the sentence *The P3 passed the contact* — shown in Fig. 9(a) — has two

```
<DefPred>      ::= "(defpred" <*Pred> <FrameMapping>+ ")".
<DefNpred>     ::= "(defnpred" <*Npred> <FrameMapping>+ ")".
<FrameMapping> ::= "(" <*FrameClass> <SlotMapping>* ")".
<SlotMapping>  ::= <*SlotName> "(" <SyntaxRole>+ "&" ")".
<SyntaxRole>   ::= "S" | "IO" | "O" | <*Role>.
```

Fig. 7 — **defpred/defnpred** macro syntax

*Abbreviation for *Loosefoot*, the voice call sign of an antisubmarine-warfare aircraft.

```

(defpred DEPART (depart :theme (S) :from-loc (O FROM &) :goal (FOR &)))
(defpred PASS
  (go-past :agent (S) :at-loc (O BY &))
  (transfer-data :agent (S) :theme (O) :to-poss (IO TO)))
(defpred RELAY (transfer-data :agent (S) :theme (O) :to-poss (TO &)))
(defnpred P3 (p-3c))
(defnpred INTELLIGENCE (data))
(defnpred INFORMATION (data))
(defnpred AX (asw-commander))
(defnpred ASWC (asw-commander))
(defnpred CONTACT
  (mutual-contact :agent (BY &) :theme (WITH &))
  (detect :theme (ON &)))
  (detected-craft)
  (detection-data))
(defnpred DEPARTURE
  (depart :theme (OF AN-STG &) :from-loc (FROM &) :to-loc (FOR &)))

```

Fig. 8 — Sample mapping definitions for RAINFORMs

SimpTerm operands, so TINSEL considers them to be implicitly role-marked S and O, respectively.*

For a frame to successfully interpret a regularization, each operand of the regularization must fill one slot of the frame, and no frame slot may be filled more than once. An ampersand at the end of a slot mapping signifies that the argument is optional for that particular Pred/Npred: selection will still succeed even if the slot is not filled. Hence, the obligatory **:theme** when mapping RELAY to predicate **transfer-data** will cause the interpreter to reject the sentence *#The P3 relayed*, but the optional **:at-loc** when mapping PASS to **go-past** permits the elliptical *The P3 passed []*.

The mappings for PASS exhibit polysemy: depending on syntactic environment, PASS can either map to **go-past** (*The P3 passed the carrier*) or **transfer-data** (*The P3 passed the carrier the info*). Finally, the pairs INFORMATION/DATA, AX/ASWC, and PASS/RELAY exhibit synonymy since they can each map to the same predicate. Even so, the syntactic mappings for PASS/RELAY are not identical, to account for such differences in behavior as *The P3 relayed/#passed the info* and *The P3 passed/#relayed the carrier the info*.

Top-Down Interpretation

Frame definitions and mapping definitions are all TINSEL needs to execute selection and interpretation of most PROTEUS regularizations. As illustration, we now step through the interpreter's behavior when it is run top-down (that is, postparse) on the two fundamental SOOP types, Wffs and Quants (i.e., regularized clauses and NPs). The Wff we are interpreting is the one shown in Fig. 9(a), and the Quants we are interpreting are its two operands. Since Quant interpretation is the simpler case, we begin there.

Interpreting Quants

The quantified expression (THE Y1 P3 SINGULAR) is the PROTEUS regularization of the noun phrase *the P3*. When that form is input to TINSEL, the interpreter first consults its internal

*Alternatively, the PROTEUS grammar writer is free to regularize any of these operands with explicit Roles of his or her choosing.

(a) Uninterpreted Wff [<i>The P3 passed the contact</i>]: (PAST PASS (THE Y1 P3 SINGULAR) (THE Y2 CONTACT SINGULAR))
(b) Unambiguous Quant interpretation [<i>the P3</i>]: (THE Y1 (:class p-3c) SINGULAR)
(c) Ambiguous Quant interpretation [<i>the contact</i>]: (ONEOF (THE Y2 (:class mutual-contact) SINGULAR) (THE Y2 (:class detect) SINGULAR) (THE Y2 (:class detected-craft) SINGULAR) (THE Y2 (:class detection-data) SINGULAR))
(d) Wff ready for selection: (PAST PASS (S (THE Y1 (:class p-3c) SINGULAR)) (O (ONEOF (THE Y2 (:class mutual-contact) SINGULAR) (THE Y2 (:class detect) SINGULAR) (THE Y2 (:class detected-craft) SINGULAR) (THE Y2 (:class detection-data) SINGULAR))))
(e) Unambiguous Wff interpretation: (PAST Y3 (:class go-past) (:agent (THE Y1 (:class p-3c) SINGULAR)) (:at-loc (THE Y2 (:class detected-craft) SINGULAR)))

Fig. 9 — Stages of semantic interpretation

(procedural) model of the SOOP syntax to determine to which SOOP type the regularization belongs. Identifying it as a Quant, the interpreter next retrieves all the `defnpred` frame mappings (Fig. 8) the user has provided for the form's `Npred`, `P3`. There is only one, a mapping to the frame **p-3c**. Since this particular form has no modifiers, interpretation is complete. TINSEL transforms the regularization into a semantic interpretation by replacing the `Npred` by a `:class` slot containing the frame name, yielding the form shown in Fig. 9(b).

Given the Quant (THE Y2 CONTACT SINGULAR), however, the interpreter finds four mappings for the `Npred` CONTACT: **mutual-contact** (as in *We had contact with relief*), **detect** (*We had a contact on a target*), **detected-craft** (*The contact submerge*), and **detection-data** (*Passed contact to relief*). Because the form is semantically ambiguous and only one of these predicates is the intended interpretation, TINSEL makes four copies of the form, inserting a different `:class` name in each, and wraps the four interpretations in the operator `ONEOF`, yielding the form shown in Fig. 9(c).

Interpreting Wffs

We can now examine how TINSEL analyzes the complete Wff of Fig. 9(a) in top-down mode. The interpreter performs the following sequence of operations.

1. TINSEL begins by recursively invoking itself (through a TINSEL function named `get-semantics`) on each of the two operands of the Wff, transforming them into interpretations as described in the previous subsection. Since interpretations are found for both Terms, TINSEL explicitly role-marks the interpretations as S and O, respectively, yielding

the form shown in Fig. 9(d).

2. Next, the interpreter locates the Wff's Pred PASS and retrieves the two defpred frame mappings for that Pred. They are

```
(transfer-data :agent (S) :theme (O) :to-poss (IO TO))
(go-past :agent (S) :at-loc (O BY &)).
```

The interpreter begins by considering the first frame mapping. According to that mapping, the only way in which the Terms of the Wff can be mapped to the arguments of **transfer-data** is from S to :agent and from O to :theme.

3. TINSEL retrieves the **transfer-data** frame definition the user has specified —

```
(transfer-data :agent org :theme data :to-poss org)
```

— to see if the frame's selectional constraints will permit this mapping to succeed. The constraint on the predicate's :agent is **org**, and the :class of the S Term is **p-3c**. Consulting the type hierarchy (Fig. 5), the interpreter confirms that **p-3c** is a subclass of **org** (via **aircraft** and **craft**), so the first Term is accepted.

4. The selectional constraint on the :theme is **data**. The O Term of the Wff is ambiguous (its class is a ONEOF set), but one member of the ambiguity set has class **detection-data**, which is a subclass of **data** in the hierarchy. The Wff has thus successfully matched **transfer-data**, but with one notable exception: an obligatory case :to-poss remains unfilled. Since no unmatched Terms remain to fill that argument, and there is no other way of mapping from the Wff to **transfer-data**, the Wff cannot be interpreted as that predicate, and the interpretation is rejected.
5. TINSEL now considers the second frame mapping, to predicate **go-past**. Again, there is only one mapping, from S to :agent and O to :at-loc. Retrieving the **go-past** frame definition

```
(go-past :agent craft :at-loc physobj),
```

the interpreter notes that the S **p-3c** is a **craft**, matching the :agent. Also, one member of the O ambiguity set is of type **physobj** (**detected-craft**), thus matching the :at-loc. Because all Terms have been accounted for and no obligatory arguments remain unfilled, a single (unambiguous) interpretation for the Wff has been found.

6. TINSEL replaces the Pred by the :class of the matching frame, replaces the syntactic role markers S and O by the thematic roles :agent and :at-loc, respectively, prunes out the rejected members of the :at-loc's ambiguity set, and generates a unique variable for the Wff for coindexing sentence adjuncts (discussed in a later section). Figure 9(e) shows the final interpretation.

Bottom-Up Interpretation

The only difference in behavior when TINSEL runs bottom-up (interleaved with the parser) is that the interpreter generally does not need to invoke itself recursively on embedded forms. This is because PROTEUS composes the regularizations of parent nodes from the regularizations of their chil-

dren, and TINSEL has already transformed the latter into interpretations earlier in the parse.* During parsing of the sentence *The P3 passed the contact*, for instance, the first node that PROTEUS submits to the selection component is the NP *the P3*, followed by the NP *the contact*, which TINSEL modifies into the interpreted regularizations seen in Figs. 9(b) and (c), respectively. When later in the parse a parent node adopts these NPs, PROTEUS composes the parent's Wff from the childrens' modified regularizations. Since both Terms of the resulting Wff have :class fields (signifying that they have already received semantic interpretations), the get-semantics function does not need to invoke the interpreter recursively on them. Role-marking the Terms yields the same form as in Fig. 9(d), ready to be submitted directly to Wff interpretation.

This definition of get-semantics allows TINSEL to run top-down, bottom-up, or a mixture of both in a data-driven rather than mode-bound manner. This is useful because interleaved interpretation cannot always proceed purely bottom-up. During parsing of the sentence *The diesel was difficult to start*, for instance, TINSEL postpones interpretation of the adjectival infinitive *difficult to start* until its subject becomes available later in the parse. This results in the partially interpreted regularization

```
(PAST DIFFICULT
  (INF START ANYONE (THE Y1 (:class diesel-engine) SINGULAR)))
```

[i.e., "it was difficult for someone to start the engine"] that contains an uninterpreted operand, the START Wff, requiring top-down analysis. That Wff's object operand already has an interpretation, however, so the top-down mode of analysis ends there.

When the interpreter is run postparse, it has no way of knowing whether a component of a regularization it is currently analyzing was also a component of a regularization it analyzed previously, so it may do redundant work. But when TINSEL is run during parsing, each Wff or Quant PROTEUS generates is only subjected to interpretation once. This efficiency, combined with the search-space pruning discussed next, make interleaved use of the interpreter particularly desirable.

Pruning the Search Space

In addition to the semantically well-formed analyses for which TINSEL can find one or more interpretations, the parser usually generates many incorrect candidates. Each of these that enters the parser's working space can spawn additional spurious nodes, expand the search, and increase processing time. By the same token, each semantically anomalous node that a selection component prunes out can prevent many other bad nodes from being generated. If selection is done as part of structure building (interpretation) as it is in TINSEL, interleaving the interpreter with the parser is beneficial only if the time saved exploring the search space outweighs the time spent constructing interpretations.

When run on one of our grammars without the interpreter, PROTEUS generates ten parses for the sentence *P3 passed contact to relief*, taking 15.9 seconds of processor time (on a Sun 3/260) and exploring a search space of 2040 nodes (Table 4, sentence A). It takes TINSEL 0.3 second to filter through all ten regularizations postparse to generate an interpretation for the correct one, resulting in a processing time of 16.2 seconds. However, when the same sentence is run with the interpreter executing selection during parsing, one parse (with interpretation) is generated after a total processor time of 12.6 seconds and a search space of 1806 nodes. This represents an 11% reduction in the search space and 22% reduction in processing time.

*Since the goal in life of the PROTEUS translation is to grow up to be logical form, we feel no qualms about discarding the former during interleaved interpretation. TINSEL will, however, also retain a copy of the normal form for debugging purposes if the user so desires.

Table 4 — Processing Time for Two Sample Sentences

sentence	A		B	
TINSEL mode	post	mid	post	mid
search space, nodes	2040	1806	21,156	3888
number of parses	10	1	491	1
parsing time, sec.	15.9	---	401.3	---
TINSEL time	0.3	---	29.8	---
total time	16.2	12.6	431.1	36.3

As the ambiguity of the input increases, so do the savings. The sentence *Passed ctc to relief via NESTOR and departed area at 0100z* (sentence B) generates 491 parses without TINSEL, because of the ambiguity caused by conjoining, numerous prepositions, the elision of subjects and determiners, and the need to accommodate possibly zeroed *be*-verbs. The parser explores a search space of 21,156 nodes and takes about 400 seconds of processor time. TINSEL takes 30 seconds to filter through the nearly 500 parses top-down to find the correct one. With TINSEL interleaved, the search space is reduced to 3888 nodes and the transpired processor time is 36 seconds, a search space reduction of over 80% and time savings of over 90%. Clearly, the time the interpreter spends constructing interpretations is minimal compared to the amount of time that selection saves the parser in node building.

Representing Global Ambiguity

The interpretation of *The P3 passed the contact* presented earlier demonstrated how the local semantic ambiguity of the words *pass* and *contact* was resolved by TINSEL's enforcement of syntactic and selectional constraints over the entire clause. If the sentence still proves ambiguous after selection, however, TINSEL returns a list of the alternative interpretations wrapped in the ONEOF operator. For example, if the *:to-poss* argument in the mapping from PASS to **transfer-data** were made optional, TINSEL would be forced to interpret *The P3 passed the contact* as ambiguous, yielding

```
(ONEOF
  (PAST Y2 (:class go-past)
    (:agent (THE Y1 (:class p-3c) SINGULAR))
    (:at-loc (THE Y2 (:class detected-craft) SINGULAR)))
  (PAST Y2 (:class transfer-data)
    (:agent (THE Y1 (:class p-3c) SINGULAR))
    (:theme (THE Y2 (:class detection-data) SINGULAR)))).
```

Paraphrase Normalization

Finally, the mapping of PASS/RELAY, INTELLIGENCE/INFORMATION and ASWC/AX to common predicates allows (1, 2) to receive identical interpretations as **transfer-data** events:

- (1) The P3 passed ASWC the info.
- (2) The P3 relayed the intelligence to AX.

(1) is accepted because the mapping from PASS to **transfer-data** allows an indirect object (ASWC) to fill the *:to-poss* argument, and in (2) the mapping from RELAY to **transfer-data** allows a TO RoleTerm to fill the same argument. In both cases, the resulting interpretation is

```
(PAST Y4 (:class transfer-data)
  (:agent (THE Y1 (:class p-3c) SINGULAR))
  (:theme (THE Y2 (:class data) SINGULAR))
  (:to-poss (THE Y3 (:class asw-commander) SINGULAR)))
```

FORMAL OVERVIEW

The preceding section presents an introduction to the fundamentals of TINSEL. We now examine the interpreter in more detail, beginning with a formal description of its input and output structures and a synopsis of the complete set of TINSEL definition macros that inform the interpreter how to transform each input structure into an output structure.

Input Syntax

The version of SOOP that TINSEL accepts (dubbed RTSOOP, with syntax shown in Fig. 10) is based on an example PROTEUS grammar named *RT* and includes several changes (in boldface) to the original definition of Fig. 2. These are the addition of adjunctive modifiers, Wff SimpTerms, and conjoined SimpTerms, and the replacement of Amods (adjective modifiers) by AnStg (adjective-noun string) modifiers. The first three are discussed in the section "Interpretation of Wffs," and the last is discussed in the section "Interpretation of SimpTerms."

<Wff>	::= "(" <*Op1> <*Pred> <Term> <*Adjuncts> ")"
<Wff>	::= "(" <*Op2> <Wff> <Wff> ")"
<Wff>	::= "(lambda (" <*Var> ")" <Wff> ")"
<Term>	::= <SimpTerm> <RoleTerm>
<Adjuncts>	::= "(S-A" <RoleTerm>+ ")"
<SimpTerm>	::= <*Con> <Quant>
<SimpTerm>	::= <Wff> "(" <*Op2> <SimpTerm> <SimpTerm> ")"
<RoleTerm>	::= "(" <*Role> <SimpTerm> ")"
<Quant>	::= "(" <*Q> <*Var> <*Npred> <*Number> <Mod> [<AnStg>] ")"
<Mod>	::= <RoleTerm> <Wff>
<AnStg>	::= "(AN-STG (" <AnMod>+ "))"
<AnMod>	::= <AMod> <NMod>
<AMod>	::= "(" ["PROG"] <*Pred> ")"
<NMod>	::= "(" <*Npred> <*Number> ")"

Fig. 10 — RTSOOP syntax

Output Syntax

Since TINSEL interpretations are transformations of PROTEUS regularizations, the interpreter's output has a SOOP-like syntax dubbed INSOOP (INterpreted SOOP), shown in Fig. 11. To illustrate some of these structures, the RTSOOP regularization (TINSEL input) and INSOOP interpretation (TINSEL output) of the sentence *An active sweep of area was conducted prior to that time* are respectively

```
(PAST CONDUCT ANYONE
  (SOME Y1 SWEEP SINGULAR
    (AN-STG ((ACTIVE))) (OF (NULL-DET Y2 AREA SINGULAR)))
  (S-A (BEFORE (THAT Y3 TIME SINGULAR))))
```

<Term>	::= <WffTerm> <QuantTerm> <ConstTerm> <VarTerm>.
<WffTerm>	::= <Wff> "(ONEOF" <Wff> <Wff>+ ")".
<Wff>	::= "(" <*Op1>+ <*Var> <ClassSpec> <Arg>+ <WffMod>* ")".
<Wff>	::= "(" <*Op2> <Wff> <Wff> ")".
<ClassSpec>	::= "(:class" <*FrameClass> ")".
<Arg>	::= "(" <*SlotName> <Term> ")".
<WffMod>	::= <RolePred>.
<RolePred>	::= "(" <*RoleClass> <VarTerm> <WffTerm> ")".
<QuantTerm>	::= <Quant> "(ONEOF" <Quant> <Quant>+ ")".
<Quant>	::= "(" <*Q> <*Var> <ClassSpec> <*Number> <Arg>* <QuantMod>* ")" "(" <*Op2> <Quant> <Quant> ")".
<QuantMod>	::= <RolePred> <Wff>.
<ConstTerm>	::= <Const> "(ONEOF" <Const> <Const>+ ")".
<Const>	::= "(const" <*Var> <ClassSpec> <*Con> ")".
<VarTerm>	::= <*Var>.

Fig. 11 — INSOOP syntax

```
(PAST Y4 (:class execute)
  (:agent (CONST Y5 (:class org) ANYONE))
  (:theme (SOME Y1 (:class minesweep) SINGULAR
    (:patient (NULL-DET Y2 (:class region) SINGULAR)
      (Y6 (:class active) (:theme Y1))))
  (:before-time Y4 (THAT Y3 (:class moment) SINGULAR))).
```

Here **execute**, **org**, **minesweep**, **region**, **active**, and **moment** are frame classes, :agent and :theme are slot names, and :before-time is a role class. The INSOOP constructs that have not yet been discussed are Wff modifiers (*prior to that time*), constants (*ANYONE*), Quant modifiers (*active*), and Quant arguments (*of area*). The first of these is examined in the section "Interpretation of Wffs," and the last three are examined in the section "Interpretation of SimpTerms."

Synopsis of TINSEL Macros

Figure 12 shows a set of TINSEL definitions that could be used to generate the semantic interpretation illustrated in the previous subsection. We have not yet covered four macros: *defq* (define quantifier), *defrole* (define role), *defadjunct* (define adjunct), and *defcon* (define constant). The last two are discussed in the next two sections respectively; we discuss the first two here.

The macros *defq* and *defrole* inform TINSEL of the Qs (Quantifiers) and Roles being used, so that the interpreter can correctly recognize the SOOP type of the forms containing them. As a convenience to the user, however, TINSEL consults the PROTEUS lexicon whenever possible to surmise whether a particular symbol is a Q or Role, so that *defq* and *defrole* only need to declare Qs and Roles that are not isomorphic with the determiners, prepositions, and subordinating conjunctions they regularize. For example, since the word *of* is a preposition, TINSEL concludes that the translation OF is a Role. Similarly, since the word *that* is both a subordinating conjunction and a determiner, TINSEL concludes that the translation THAT can be either a Role or a Q (the latter is the case in our example). But *SOME*, *NULL-DET*, and *BEFORE* (translations of the words *a*, the zeroed determiner, and the idiom *prior to*) are not identical to the words they regularize, so (if the words *some* and *before* are not in the lexicon) TINSEL must be informed which SOOP functor each translation represents.

```

(defframe execute :isa process :agent org :theme process)
(defframe minesweep :isa process :patient region)
(defframe active :isa state :theme minesweep)
(defframe region :isa location)
(defframe moment :isa time-unit)
[also frame definitions of process, org, state, etc.]

(defpred CONDUCT (execute :agent (S) :theme (O)))
(defnpred SWEEP (minesweep :patient (OF &)))
(defpred ACTIVE (active :theme (S)))
(defnpred AREA (region))
(defnpred TIME (moment))

(defq NULL-DET SOME)
(defrole BEFORE)

(defadjunct BEFORE (:before-time process moment))
(defcon ANYONE org)

```

Fig. 12 — Overview of TINSEL macros

INTERPRETATION OF WFFS

We now look in more detail at TINSEL's interpretation of verbal and adjectival predicate-argument structures, or Wffs.

Multiple Frame Mappings

A `defpred` mapping can capture a wide variety of verb-operand behaviors, but a single mapping is sometimes not sufficient. For instance, consider the following possible occurrences of the verb *bake* in ordinary English:

- (1) Nadia was baking the brownies.
- (2) The brownies were baking.
- (3) The oven was baking the brownies.
- (4) The brownies were baking in the oven.
- (5) #Nadia was baking in the oven.
- (6) #The oven was baking.
- (7) Nadia was baking.
- (8) Nadia was baking Ross the brownies.
- (9) Nadia was baking the brownies for Ross.
- (10) #The brownies were baking for Ross.
- (11) #Nadia was baking for Ross.

Figure 13(a) shows a first cut at a mapping definition for this verb. This definition accounts for (1-6), but since the `:patient` has been made obligatory to disallow (5,6), the well-formed (7) will not pass selection. Furthermore, if we were to add a `:beneficiary` role to account for (8,9), we would want to constrain such sentences to always have both an `:agent` and a `:patient` to reject (10,11), which the definition as it now stands could not enforce.

In TINSEL, complex verb behavior like this is modeled by providing the `defpred` statement with multiple frame mappings. The new definition of Fig. 13(b) enforces the behaviors just outlined. Each frame mapping of the new definition matches a different subset of the sample data. The first mapping continues to match (1-6), and the second mapping nicely captures the rule that the `:patient`

(a) Initial mapping definition:
(defpred BAKE (oven-cook :agent (S &) :patient (S O) :at-loc (S IN &)))
(b) Extended mapping definition:
(defpred BAKE (oven-cook :agent (S &) :patient (S O) :at-loc (S IN &)) (oven-cook :agent (S)) (oven-cook :agent (S) :patient (O) :beneficiary (FOR IO) :at-loc (IN &)))

Fig. 13 — Modeling complex verb behavior

can be omitted only when there is an :agent and no :at-loc, and so accounts for (7). The third mapping handles the co-occurrence constraints on :beneficiary by requiring that both the :agent and :patient be filled when the :beneficiary is filled (i.e., all three roles are obligatory), thus accepting (8, 9) and rejecting (10, 11).

Interaction with Strict Subcategorization

Strict subcategorizations are constraints on the types of syntactic structures acceptable as complements to a verb. In PROTEUS, such constraints are coded in the lexicon and enforced by the parser as grammatical restrictions. Subcategorizations often constrain the regularizations with which TINSEL defpreds must deal. For instance, the verb *bake* might have a subcategorization allowing its complement list to be empty, as in (2, 6, 7) of the sample data: *an NP*, as in (1, 3); a PP headed *in*, as in (4, 5); an NP followed by a PP headed *in* or *for*, as in (9); or two NPs, indirect and direct object, as in (8). In each of these cases, after the (syntactic) subcategorization constraint has been enforced, TINSEL then steps in to check the complement's semantic validity. The user, however, might choose to rule out (10, 11) on lexical/syntactic grounds (without the parser having to consult the interpreter at all) by not allowing *bake* to subcategorize for a complement consisting simply of a *for* PP. Such a subcategorization does not appear to allow the defpred of Fig. 13(b) to be rewritten in a simpler form, but in other cases it might simplify the coding of the verb's mapping definition. Strict subcategorization can thus be viewed as a relatively inexpensive filtering operation that can reduce the amount of semantic interpretation done.

Since PROTEUS is a syntax-driven system in which "grammar proposes and semantics disposes" (unlike a semantics-driven *case frame parser* in which predicate-argument models actively build analyses directly from the sentence string), TINSEL can produce interpretations only if the parser furnishes it with correct regularizations. Thus, care must be taken that the strict subcategorizations of the verbs in the lexicon are adequate to cover the data. If subcategorization incorrectly rejects a complement to a particular verb, the parse is discarded without any consultation of the interpreter. For example, if *bake* is not subcategorized for a complement string consisting of two NPs followed by an *in* PP (*Nadia was baking Ross some brownies in her new oven*), the parse is rejected as syntactically ill-formed, and TINSEL is never offered a regularization to interpret.

Depassivization

When PROTEUS encounters a passive sentence containing a verb-complement PP headed *by*, TINSEL expects that a parse will be generated that regularizes the PP's complement into subject position. Hence, *The horse was raced by the owner* should generate the two parses

- (a) (PAST RACE ANYONE (THE Y1 HORSE SINGULAR)
(BY (THE Y2 OWNER SINGULAR)))
- (b) (PAST RACE (THE Y2 OWNER SINGULAR) (THE Y1 HORSE SINGULAR)),

corresponding to the two analyses of *by* as locative and agent. The reason for this approach is that the dummy subject *ANYONE* distinguishes the regularizations of agentless passives (*The horse was raced*) from objectless sentences (*The horse raced*), but regularization (a) cannot represent the agentive passive because two different Terms of a Wff cannot both fill the same semantic role.*

Regularization of Predicate Adjectives

Our PROTEUS lexicons use raising operators to regularize a sentence like *The door is open* into Wff (a), which should be interpreted as a unary statal predicate.

- (a) (PRESENT OPEN (THE Y1 DOOR SINGULAR))
- (b) (PRESENT PROG CALCULATE NADIA)
- (c) (PRESENT COMPLICATE (THE Y1 PUMP SINGULAR))

This Wff, however, is identical to the default regularization of the verbal *The door opens*, which should be interpreted using a relational predicate (e.g., *The door opens from the inside with a key*). The user must thus be careful that the root form OPEN is not used as the translation for both the adjective and the verb. The same is true of participial adjectives: *Nadia is [extremely] calculating* and *The pump is complicated* cannot be allowed to regularize respectively as Wffs (b,c), because (b) is indistinguishable from the verbal *Nadia is [busily] calculating*, and (c) is indistinguishable from the ill-formed *#The pump complicates*.

Adjuncts

Regarding the difference between verb complements and modifiers, Gawron [8] states

Note that [SOOP] passes over an important linguistic distinction, the distinction between argument (or a constituent dominated by OBJECT, in LSP terms) and adjunct. Both can wind up in the same place in the op/op representation...This choice does not reflect any belief that there IS no semantic distinction between arguments and adjuncts; rather it is made because this representation is designed to feed selection, and selection, in general, seems the best place to make the decision between calling something an argument and calling it an adjunct.

Although it is true that selection is the place for the final decision to be made, the grammars our project uses generate alternative parses depending on whether a Term is being hypothesized as an argument or adjunct. Selection's job is to then choose which of those syntactic analyses is correct. Rather than ignore the operand/adjunct structural distinction the parser enforces, RTSOOP regularizes sentence adjuncts with the distinguished marker S-A, while continuing to regularize verb complements as Wff operands (ordinary SimpTerms and RoleTerms). For example, the sentence *Virginia departed from the area at noon* would be regularized as

(PAST DEPART VIRGINIA (FROM (THE Y1 AREA SINGULAR)) (S-A (AT NOON))).

TINSEL requires that Wff operands (here, *from the area*) be interpretable only as predicate arguments, and adjuncts (*at noon*) only as binary RolePreds (role predicates) modifying the host Wff. Mappings from adjunct to RolePred are encoded by use of the TINSEL macro *defadjunct*. For example, based on the definitions

*As a result of having PROTEUS execute this transformation, *defpred* expressions need only contain agentive mappings from S (subject) and never from BY.

```

(defadjunct AT
  (:at-time process (hour-expression moment))
  (:at-loc process location))
(defadjunct ON
  (:at-time process day-expression)
  (:at-loc process location))
(defadjunct WHEN (:at-time process process))

```

the sentence *Lsft departed on Tuesday at 0100z when relief arrived* would receive the following PROTEUS regularization and TINSEL interpretation, respectively:

```

(PAST DEPART
  (NULL-DET Y4 LOOSEFOOT SINGULAR)
  (S-A (ON TUESDAY)
    (AT (NULL-DET Y2 Z SINGULAR (VALUE 100)))
    (WHEN (PAST ARRIVE (NULL-DET Y3 RELIEF SINGULAR))))))

(PAST Y1 (:class depart)
  (:agent (NULL-DET Y4 (:class asw-aircraft) SINGULAR))
  (:at-time Y1 (CONST Y5 (:class day-expression) TUESDAY))
  (:at-time Y1 (NULL-DET Y2 (:class hour-expression) SINGULAR
    (:value 100)))
  (:at-time Y1 (PAST (:class arrive)
    (:agent (NULL-DET Y3 (:class relief-craft)
      SINGULAR)))))).

```

Whereas a predicate argument is selected by its host verb, a RolePred selects both its host and complement. Also unlike predicate arguments, the same RolePred can occur more than once in a single interpretation. In our example, the Roles ON, AT, and WHEN are all interpreted as the RolePred `:at-time`, which takes the host predicate **depart** as its first argument (by coindexing) and various time or event tokens as its second argument. AT can be mapped to `:at-time` if its host predicate is a **process** (e.g., **depart**) and its complement is either an **hour-expression** or a **moment** (*Lsft departed at that time*). ON, however, selects for a **day-expression** complement (*Tuesday*), and WHEN selects for a **process** complement (*relief arrived*).

RolePreds correspond to what Allen calls "outer cases," as distinguished from "inner cases" or predicate arguments. Outer cases, typically time, place, or manner descriptors, are frequently adverbial and can occur with a wide variety of host verbs. Otherwise, they are logically equivalent to predicate arguments: for instance the RolePred structure `(:at-time Y1 x)` can be considered a longhand notation for the argument form `(:at-time x)`.

Wff SimpTerms

Although the original SOOP syntax does not cover it, there are several circumstances in which it appears that Wffs should be permitted to act as SimpTerms (i.e., non-Role-marked operands). First, subordinate clauses like *when relief arrived* regularize as RoleTerms, which means that the Wff *relief arrived* must be a SimpTerm. We saw in the previous section how TINSEL interprets such clauses using RolePreds.

Second, high-order predicates like *easy* and *want* that can take infinitival or progressive complements regularize as Wffs that have Wff operands. For example, our PROTEUS grammars for TINSEL

*RolePred notation also provides a correspondence between frame notation and first-order predicate logic, e.g., `departure(Y1) & agent(Y1, Y4) & asw-aircraft(Y4) & at-time(Y1, Y5) & ...`

recognize the classic distinction between the predicate adjectivals

- (1) Nadia is easy to please.
- (2) Nadia is eager to please.

by regularizing them as unary and binary forms respectively:

- (a) (PRESENT EASY (INF PLEASE ANYONE NADIA))
- (b) (PRESENT EAGER NADIA (LAMBDA (X) (INF PLEASE X))).

Here, a PLEASE Wff is the first operand of form (a) and the second operand of form (b). The following definitions will successfully interpret these forms:

```
(defframe satisfy :isa process :agent person :patient person)
(defframe simple :goal process)
(defframe enthusiastic :experiencer person :goal process)

(defpred PLEASE (satisfy :agent (S) :patient (O &)))
(defpred EASY (simple :goal (S)))
(defpred EAGER (enthusiastic :experiencer (S) :goal (O &))).
```

The **process** selection constraint on the :goal of each predicate allows both **action** (e.g., *Nadia is eager to please*) and non-**action** processes (*Nadia is eager to receive the package*) to be accepted as goals.

Note that the infinitival argument of (b) is a lambda form having an unbound subject variable X. When TINSEL sees a lambda form, it binds the variable to the semantic interpretation of the immediately preceding *SimpTerm* before undertaking the Wff's interpretation. Hence, in (b) it correctly analyzes the subject of the embedded PLEASE Wff as NADIA. Furthermore, the binding of the variable takes place only after the *SimpTerm* has first passed selection against its own host predicate. For example, in the sentence *The brownies were eager to bake*, all three interpretations of BROWNIE (**cookie**, **gnome**, and **girl-scout**) are acceptable as the subject of *bake*, but by first testing *the brownies* as the subject of *eager*, the **cookie** interpretation is discarded.

TINSEL's input should also distinguish binary verbal forms like (3, 3') from ternary ones like (4) by regularizing them respectively as (c, c') and (d):

- (3) Ross wanted to whistle.
 - (3') Ross wanted Nadia to whistle.
 - (4) Ross ordered Nadia to whistle.
- (c) (PAST WANT ROSS (LAMBDA (X) (INF WHISTLE X)))
 - (c') (PAST WANT ROSS (INF WHISTLE NADIA))
 - (d) (PAST ORDER ROSS NADIA (LAMBDA (X) (INF WHISTLE X))).

Here we might want a predicate model like the following:

```
(defframe whistle :isa action :agent person)
(defframe desire :experiencer person :goal process)
(defframe command :agent person :to-poss person :goal action)

(defpred WHISTLE (whistle :agent (S)))
(defpred WANT (desire :experiencer (S) :goal (O)))
(defpred ORDER (command :agent (S) :to-poss (IO) :goal (O))).
```

Note that we tighten the selection constraint on the :goal of the **command** predicate to **action**, to reject *#Ross ordered Nadia to receive the package*. Again, since the lambda variable gets bound to the immediately preceding SimpTerm, in (c) the subject of WHISTLE is ROSS, and in (d) it is NADIA. If there are recurrent nested lambdas, as in the regularization of *Ross enjoyed ordering Nadia to tell Ralph to whistle*—

```
(PAST ENJOY ROSS
  (LAMBDA (X) (PROG ORDER X NADIA
    (LAMBDA (X) (INF TELL X RALPH
      (LAMBDA (X) (INF WHISTLE X)))))))
```

— each variable receives a different local binding: the subject of ORDER is bound to ROSS, the subject of TELL to NADIA, and the subject of WHISTLE to RALPH, yielding the proper interpretation.

Finally, when a progressive (i.e., gerund) or infinitival form occupies or has been extraposed from subject position (*Flying planes is dangerous*, *To fly planes is dangerous*, *It is dangerous flying planes*, *It is dangerous to fly planes*), there is no SimpTerm to which to bind the lambda variable. Since TINSEL postpones the interpretation of lambda forms until a subject becomes available, it currently requires that such forms regularize instead as conventional Wffs with a dummy Con (constant) as subject, for example

```
(PRESENT DANGEROUS (PROG FLY ANYONE (NULL-DET Y1 PLANE PLURAL))),
```

where ANYONE is a Con. The specification and interpretation of constants are discussed shortly.

Conjoined Operands

TINSEL requires that all members of a conjoined SimpTerm pass selection as the same argument of their host predicate. For example, in the Navy CASREP (Casualty Report) sentence *Low lube oil and fail to engage alarms sounded*, TINSEL rejects the bad parse "oil and alarms sounded" because, although *alarm* is acceptable as the :theme of predicate **sound**, *oil* is not. A more interesting example is the sequence of sentences

- (1) Nadia was baking.
- (2) The cookies were baking.
- (3) #Nadia and the cookies were baking.

where in (1,2) each NP individually plays an acceptable thematic role as the subject of *bake* (*Nadia* is :agent and *the cookies* is :patient), but the interpreter rejects (3) because those roles are not the same. Similarly, *#Nadia opened the door and a new business* is rejected because a single interpretation of the polysemous verb *open* does not accept both members of the object, although different interpretations would accept each operand individually.

INTERPRETATION OF SIMPTERMS

Next we look closer at the interpretation of the two fundamental types of SimpTerms, Quants (i.e., determinate noun phrases) and Cons (proper names and pronouns). We begin with a discussion of TINSEL's treatment of underspecific SimpTerms, followed by its handling of Cons and Quant modifiers. The section closes with a discussion of selection as an approximate solution to the general problem of denotation identification.

Underspecification

Ordinarily a *SimpTerm* will pass a domain constraint only if the class of its interpretation is a subclass of (or the same as) the constraint class. So if the *:theme* of a predicate *fly* has selectional constraint *aircraft*, arguments of type *aircraft*, *helicopter*, and *jet* will all pass selection. Similarly, if the *:theme* of the predicate *hover* has the tighter constraint *helicopter*, an argument of type *jet* is rejected, not being below *helicopter* in the sort hierarchy. *The aircraft hovered* is semantically acceptable, however, because *the aircraft* might be an underspecific reference to a helicopter. So TINSEL also permits an interpretation to pass selection if its class is an ancestor of the domain constraint, as *aircraft* is to *helicopter*. Ultimately, of course, underspecified NPs must be resolved by dereferencing in cooperation with a discourse component. This TINSEL mechanism allows such NPs to pass selection even though their precise referents have not yet been determined.

Constants

TINSEL provides a macro called *defcon* (define constant) with which the user specifies the semantic class of constants, allowing the interpreter to expand them into Quant-like interpretations. For example, the definition

```
(defcon NADIA person)
```

allows the Con (constant) NADIA to be interpreted as

```
(CONST Y1 (:class person) NADIA),
```

providing a semantic class for selection purposes as well as an identifier variable for possible coindexing needs. The symbol *ANYONE* that the example *PROTEUS* grammars provide as the subject operand of agentless passives is a Con that appears frequently, so *defcon* can be used to give it an agentive semantic class (e.g., *org*). TINSEL also assumes that pronouns regularize as Cons and are assigned appropriate types, such as *female-person* for *she*, and *thing* for *it* and *they*. However, the distinguished constant *VAR* (to be discussed next) is processed as a local variable by TINSEL and so should not be assigned lexical semantic class. Finally, integer values are automatically recognized as the LISP data type *fixnum* and do not have to be declared as Cons.

Quant-modifying Wffs

So far we have considered only the interpretation of unmodified NPs. The RTSOOP syntax of Fig. 10 shows that Quants can take three different kinds of modifiers: Wffs (relative clauses), RoleTerms (PPs), and AnStgs (premodifiers). Each of these is now be discussed in turn, beginning with Wffs.

Our *PROTEUS* grammars represent *Wh*-movement by leaving the distinguished variable *VAR* as a trace of the moved *SimpTerm*. For example, *The brownie that Nadia baked* regularizes as

```
(THE Y1 BROWNIE SINGULAR (PAST BAKE NADIA VAR)),
```

where the *Wh*-tracing *VAR* occupies the object operand position of the embedded *BAKE* Wff (i.e., relative clause). To interpret this form, TINSEL begins by analyzing the *Npred* *BROWNIE* as usual, generating the *ONEOF* ambiguity set {*cookie*, *gnome*, *girl-scout*} as an interim interpretation. It then proceeds to bind *VAR* to this set of values and interpret the embedded *BAKE* Wff. Only one value, *cookie*, passes selection as the object of *bake*, so it is accepted as the *:class* of the form. *VAR* is replaced by the form's identifier variable, yielding the interpretation

```
(THE Y1 (:class cookie) SINGULAR
  (PAST Y2 (:class oven-cook)
    (:agent (CONST Y3 (:class person) NADIA))
    (:patient Y1))).
```

During parsing, TINSEL postpones interpretation of the embedded Wff until it attaches to the host form, since the value of VAR is not known until that time. Interpretation is also postponed on the arbitrary number of progressive and infinitival Wffs that can embed a *Wh*-tracing Wff, as in *the brownie that Nadia enjoyed ordering Ross to tell Ralph to congratulate*. The "Wh-Island Constraint" is enforced by rebinding VAR each time a new relative clause is entered. For example, in *What did the brownie Nadia baked taste like?*,

```
(ASKWH (WHICH Y1 THING)
  (PAST TASTE (THE Y2 BROWNIE SINGULAR (PAST BAKE NADIA VAR))
    (LIKE VAR))),
```

the complement of LIKE is bound to the long-distance host Y1 (i.e., *what*) but the object operand of BAKE is bound locally to Y2 (*brownie*).

Quant-modifying RoleTerms

Quant-modifying RoleTerms fill a variety of semantic roles. Just as Wff-modifying RoleTerms can be interpreted either as arguments or as adjuncts, a parallel seems to exist in Quants. Attached to head nouns that are relational in nature (nominalizations, quantifying nouns, and others), RoleTerms are often best modeled as arguments, but attached to nonrelational head nouns they tend to act like abbreviated relative clauses (modifiers). TINSEL tries to analyze each Quant-modifying RoleTerm as one or more of the following:

- **Argument:** of nominalization (the invention/inventor of the wheel, a conversation with the inventor), quantifier (the majority of the wheels), other (a diameter of 12 inches). The RoleTerm is interpreted as an argument of the host predicate.
- **Adjunct:** of nominalization (the invention in Mesopotamia of the wheel). The RoleTerm is interpreted as a RolePred modifier.
- **Implicit relative clause:** BE (the ruins in Mesopotamia, the invention on the shelf, the man with the inventor), HAVE (an inventor with an idea, the diameter of the wheel), other (the wheel as a tool). The RoleTerm is interpreted as a Wff modifier.

Argument

Nominalizations. Verb-derived nouns are common in the RAINFORM data and are a major source of Quant-modifying RoleTerms:

```
initial contact with LAMPS
PAX transfer to Virginia
attack on Peterson
no communications with Brumby
communications relay between Brumby and Miller
under control of Constellation
commencement of period
MAD sweeps of area
```

(and one instance, *safety of flight*, that nominalizes the adjective *safe*). In nominalizations, the OF Role typically denotes either the subject (*commencement of period*) or object relation (*sweeps of area*), and other Roles (BY, WITH, etc.) fill the same relations as in passive verbal constructions. Mapping a nominalization to the same argument-taking predicate as the corresponding verb or adjective allows TINSEL to interpret the modifying RoleTerms as arguments. For example, the definitions

```
(defframe destroy :agent org :patient physobj :instrument tool)
(defpred DEMOLISH (destroy :agent (S) :patient (O) :instrument (WITH &)))
(defnpred DEMOLITION
  (destroy :agent (BY &) :patient (OF AN-STG &) :instrument (WITH &)))
```

permit (1, 2) to receive the same interpretation:

- (1) We watched the crew demolish the building with dynamite.
- (2) We watched the demolition of the building with dynamite by the crew.
- (3) We watched the demolishing of the building with dynamite by the crew.

DEMOLITION's modifier Roles BY, OF and WITH are simply mapped to the appropriate slots of the same frame used to interpret the Pred DEMOLISH.* Note that none of the slot mappings is obligatory, since the bare NP *the demolition* is perfectly acceptable.

Sentence (3) also receives the same interpretation as (1, 2). The RT grammar regularizes the progressive verbal NP *the demolishing* as a special form intermediate between a Quant and a Wff,

(THE Y1 PROG DEMOLISH).

Since TINSEL must interpret this form by using the defpred mappings for the Pred DEMOLISH (which do not include BY and OF), it generates a copy of the mappings in which all occurrences of the Role S have been replaced by OF and BY, O by OF and AN-STG, and optional-argument markers have been introduced (to permit bare phrases like *the singing*). For example, it transforms the first mapping below into the second:

```
(NAVIGATE (pilot :agent (S) :theme (O) :instrument (BY)))
(NAVIGATE (pilot :agent (OF BY &) :theme (OF AN-STG &)
           :instrument (BY &))),
```

which will now correctly interpret *the navigating of the river by the captain*, *the navigating of the river by compass*, *the navigating of the captain*, *river navigating*, etc.

Nominalizations that refer not to the entity the verb describes but to an argument of the verb are sometimes called "role nouns." For instance the noun *singer* can be modeled as the subject of the verb *sing* rather than as a special subclass of **person**, allowing the *of* phrase in *singer of anthems* to map to the verb's object [i.e., "person that sings anthems"]. A few nouns in the RAIN-FORM data might be considered role nouns:

```
Acted as communications relay between units. ["relayer"]
Completed swap with relief. ["reliever"]
The contact submerged. ["contactee"]
```

*The Role AN-STG permits interpretation of the NP *building demolition*, discussed in the section on "NP Premodifiers."

PROTEUS could conceivably be used to regularize role nouns into Wff-modified forms, such as regularizing the NP *relay* as

```
(NULL-DET Y1 CRAFT SINGULAR (RELAY VAR ANYTHING))
```

["craft that relays something"], but this would require specialized grammar rules to process any modifiers the form might have. For example, *communications relay between units* is not a "communications craft between units that relays something," but a "craft that relays communications between units." It would also entail that the user provide multiple lexical entries for nouns that have both role-noun and other senses (e.g., the noun *relay* is also an event of relaying), which violates our philosophy of one word entry per lexical sense (part of speech). The role of the PROTEUS translation is to regularize syntax by assigning it logical-form structure, not to undertake semantic decomposition of individual lexical items.

A better approach to dealing with a role noun like *relay*, therefore, is to map it to a sort predicate **relaying-craft** (a subclass of **craft**) that has a :theme slot just like the verbal predicate **transfer-data** does (the mapping in Fig. 8 of the Npred CONTACT to **detected-craft** is an example of this approach). The modifier in *communications relay*, for example, can then be mapped to that slot, yielding

```
(NULL-DET Y1 (:class relaying-craft) SINGULAR
  (:theme (NULL-DET Y2 (:class comm-data) SINGULAR))).
```

Establishing a relationship between this expression and the **transfer-data** predicate (if appropriate) is left to later stages of semantic analysis.

Quantifying Nouns. Examples of nouns from the RAINFORM data that can be involved in quantifying (for example, partitive) constructions are

```
portion of mission
portion of area
remainder of period
starboard quarter of the formation
corner of oparea
majority of onsta tasking.
```

Quantifying nouns act more like quantifiers than they do set-specifiers, which causes problems for selection. For example, in *Crew aborted portion of mission* it is not the head noun *portion* that should be checked for selection against the host verb, but the NP's *of* complement (e.g., *#Crew aborted portion of area*). This is an oversimplification, since certain verbs do not exhibit this behavior (*#John drove a portion of his car*) and the *of*-complement's number can also influence whether the behavior occurs (*John drove most of his cars*, *#John drove most of his car*). These are denotational problems that simple lexical interpretation is not prepared to handle.

In all our RAINFORM examples of quantifying nouns, however, *of*-complement selection works correctly, so TINSEL has been hardcoded to recognize a distinguished slot named :q-of (quantifier-of) that can be used to enforce this behavior. When applying a selectional test to an interpreted Quant that has a :q-of slot, the interpreter tests not the class of the Quant itself, but the class of the value filling the slot. If the slot contains no value, as in the elliptical *Crew aborted a portion*, the test is applied to the slot's type constraint instead.

To illustrate, suppose we want to enforce the following selectional behavior in a toy domain:

- (1a) a piece/#glassful of brownie
- (1b) a piece of the sidewalk
- (1c) a portion of the brownie/soda
- (1d) several of the brownies
- (2a) Nadia ate/#drank a piece/portion of the brownie.
- (2b) #Nadia ate a piece of the sidewalk.
- (3a) Nadia ate several of the brownies.
- (3b) Nadia thanked several of the brownies.
- (4a) Nadia ate/#drank a piece.
- (4b) Nadia ate/drank a portion.
- (4c) Nadia ate/drank/thanked several.

The quantifying NPs could be modeled as follows:

```
(defframe quantity :q-of all)
(defframe phys-quantity :isa quantity :q-of physobj)
(defframe solid-quantity :isa phys-quantity :q-of solid-physobj)
(defframe fluid-quantity :isa phys-quantity :q-of fluid-physobj)
(defnpred NULL-N (quantity :q-of (OF &)))
(defnpred PORTION (phys-quantity :q-of (OF &)))
(defnpred PIECE (solid-quantity :q-of (OF &)))
(defnpred GLASSFUL (fluid-quantity :q-of (OF &))).
```

The NPs in (1a-d) are handled conventionally. In (1a,b) **PIECE** maps to **solid-quantity**, which accepts any **solid-physobj** (e.g., *brownie*, *sidewalk*) as :q-of, but **fluid-quantity** (**GLASSFUL**) only accepts a **fluid-physobj** argument like *soda*. In (1c) **PORTION** maps to **phys-quantity**, which accepts any physical entity (*brownie*, *soda*) whether solid or fluid. Finally, in (1d) the zeroed head noun regularizes as the Npred **NULL-N**, which we map to a generalized quantifying relation **quantity** that can take any class of :q-of argument.* Since *brownies* can be interpreted either as **food** or **girl-scout**, (1d) receives both interpretations. Next, in (2a) **EAT** accepts the object *a piece* (*portion*) of the *brownie* because the NP's :q-of slot contains an entity of type **food**, but *drink* will not accept a non-**fluid-physobj** argument, nor will **EAT** accept the non-**food** *sidewalk* in (2b).

In (3a) **EAT** accepts only the **food** interpretation of (1d), whereas in (3b) it is the **girl-scout** interpretation that **THANK** accepts. Finally, in (4a-c) we deal with elliptical forms in which the *of*-complement is absent. In (4a), **EAT** accepts the NP *a piece* because the selection constraint on **solid-quantity**'s :q-of slot is **solid-physobj**, which **TINSEL**'s underspecification mechanism sees as a possible underspecific reference to **food**. A *piece* cannot be seen as an underspecific reference to **beverage**, however, so **DRINK** rejects the phrase. In (4b), **phys-quantity**'s :q-of selection constraint is just **physobj**, so both **EAT** and **DRINK** accept the NP *a portion*. And in (4c), the zeroed head noun **NULL-N** is interpreted as a **quantity** with :q-of constraint **all**, which all the verbs will accept.

In (4a-c), of course, contextual analysis will be required post-**TINSEL** to resolve which particular entity in the discourse context was being referred to by each elliptical element. **TINSEL**'s selection constraints will prove useful in this phase of analysis as well, such as searching prior discourse for an instance of type **food** that the NP *a piece* in (4a) is an underspecific reference to.[†]

*We treat determiner-noun agreement, such as requiring that the *of*-complement in (1d) be a plural count noun, as a syntactic constraint enforced by the parser rather than a selectional constraint.

†Predicate models might also be useful in assembling a plausible predicate-argument structure from the parse fragments generated by a bottom-up parse of syntactically ill-formed input.

Other Relational Nouns. Several other classes of nouns in the RAINFORM data take modifiers (particularly *of*-phrases) that act like argument roles:

- (1a) area of 45 degrees
- (1b) speed of 45 knots
- (2a) sign of distress
- (2b) photo of ship
- (2c) report of ship departing channel
- (3a) north of carrier
- (3b) within 25 miles of carrier
- (3c) 5 miles from ship
- (3d) 2 hours after restart time.

In each case the head noun should be mapped to a sort predicate having an appropriate slot to hold the *of*-complement. For example, (1a) might map to a predicate **area** having a :value slot. The *of*-complement in an NP like *the area of the pattern*, however, should instead be interpreted as an NP adjunct, discussed next.

Adjunct

The interpreter analyzes outer case modifiers on nominalizations (*the departure on Tuesday*) just as it does outer case modifiers on Preds (*Usft departed on Tuesday*), by using defadjunct mappings to analyze the preposition as a binary RolePred modifier taking the host as its first argument and the complement as its second argument.

Implicit Relative Clause

Most RoleTerms that modify non-verb-derived head nouns can be paraphrased as restrictive or appositive relative clauses with covert verb *be* (*a man on the radio* can be analyzed as "a man who is on the radio"). (1-4) are examples from the RAINFORM data, many representing locative relationships.

- (1) AAWC on Virginia
- (2) area between TASS ships
- (3) barrier ahead of ship
- (4) fire of unknown origin.

In all these cases, TINSEL expands the modifier into a BE Wff, for example transforming Quant (a) — the regularization of (1) — into Quant (b):

- (a) (NULL-DET Y1 AAWC SINGULAR (ON (NULL-DET Y2 VIRGINIA SINGULAR)))
- (b) (NULL-DET Y1 AAWC SINGULAR
(BE VAR (ON (NULL-DET Y2 VIRGINIA SINGULAR)))).

TINSEL then attempts to interpret the expanded form by using predicate models for the Pred BE. BE could have a wide variety of models depending on the domain. For example, the following definitions allow *a man on the radio* to receive the two interpretations of a person appearing on a broadcast or physically atop an appliance:

```
(deframe appear-on :theme person :at-loc broadcast)
(deframe phys-upon :theme physobj :at-loc physobj)
(defpred BE (appear-on :theme (S) :at-loc (ON))
  (phys-upon :theme (S) :at-loc (ON))).
```

The Roles WITH and OF, however, often occur in contexts that cannot be paraphrased using a BE relative clause. Some examples from the RAINFORM data are

- (5) position of craft
- (6) speed of operations
- (7) area of the pattern
- (8) signature of Whiskey
- (9) grey hull with cream superstructure aft.

For example, (9) cannot be interpreted as "a hull that is with a superstructure." Instead these Roles translate to the verb *have*, a "vague predicate" with a wide variety of possible interpretations (*a man who has blue eyes/a cold/a good job...*). In such cases, WITH's complement maps to the object (*hull with superstructure* can be interpreted as "a hull that has a superstructure"), and OF's complement maps to the subject (*the superstructure of the hull* can be interpreted as "the superstructure that the hull has"). TINSEL is hardcoded to execute the appropriate translation and then attempt to interpret the resulting form using any predicate models the user has provided for the Pred HAVE.

Finally there are covert predicates other than *be*, as in *television [acting] as an instrument of propaganda* and *the tool [right] for the job*. Although these are uncommon, TINSEL executes one last expansion of each Quant-modifying RoleTerm into a Wff with the dummy Pred COVERT-PRED. If the user has provided any defpred models for COVERT-PRED, e.g.,

```
(defpred COVERT-PRED
  (act :theme (S) :role (AS))
  (right :theme (S) :goal (FOR))),
```

an attempt is made to analyze the phrase by using them.

NP Premodifiers

NP premodifiers tend to be particularly problematic in semantic interpretation, primarily because of difficulties introduced by the N-N (noun-noun) construct. An example is the RAINFORM NP *excellent NESTOR comms*, where one noun (*NESTOR*) modifies another (*comms*). Since no overt lexical predicate (i.e., verb or adjective) or even role marker (preposition) is present to suggest the semantic link between the two nouns, selection — defined here as the enforcement of lexical predicate-argument co-occurrence patterns — has nothing to work with. Instead, world knowledge and inferencing must be applied to recover the implicit relational structure "communications conducted over a NESTOR circuit."

Although the original SOOP definition regularizes adjective modifiers as Amods (Fig. 2), the RT grammar analyzes most premodifiers as a flat adjective-noun string,* regularized as a labeled list of embryonic predicate forms. For instance, the grammar regularizes *excellent NESTOR comms* as

```
(NULL-DET Y1 COMMUNICATION PLURAL (AN-STG ((EXCELLENT) (NESTOR SINGULAR)))).
```

where (EXCELLENT) can be considered an embryonic Wff — just a Pred with no operands — and (NESTOR SINGULAR) an embryonic Quant — just an Npred and Number. The grammar does not attempt to identify *excellent* as an Amod on the head noun *comms*, because the adjective might be modifying the noun *NESTOR* instead — an ambiguity produced by the N-N structure. By not explicitly

*Typical exceptions are numeric quantifiers (*24 hours*) and genitives (*Nadia's marmot*), which can be regularized as Role-Attribute pairs or RoleTerm modifiers.

generating these alternative structural analyses, the *RT* grammar recognizes that predicate-argument selection is not the appropriate mechanism for dealing with most N-N relations. Premodifier sequences can undergo selectional analysis, however, if they consist only of inner-case noun or adjectival modifiers of the head noun, as follows.

Inner Case Noun Premodifiers

If the head Npred corresponds to an argument-taking predicate, a noun modifier can often be analyzed as one of those arguments. Hence, *equipment malfunctions* can be analyzed as

```
(NULL-DET Y1 (:class equip-fail) PLURAL
  (:patient (NULL-DET Y2 (:class equip) SINGULAR)))
```

(the same way as *malfunctions of equipment*) by the TINSEL mapping

```
(defnpred MALFUNCTION (equip-fail :patient (OF AN-STG &))),
```

which tells the interpreter that the *:patient* can either be a RoleTerm marked OF, or an embryonic Quant in the AN-STG list.

Adjective Premodifiers

Adjectival premodifiers are handled by expansion into Wff modifiers. For example, since it happens that *excellent* modifies the head noun in *excellent NESTOR comms*, TINSEL can transform the PROTEUS regularization of that phrase into

```
(NULL-DET Y1 COMMUNICATION PLURAL (AN-STG ((NESTOR SINGULAR)))
  (EXCELLENT VAR))
```

[*"NESTOR communications that are excellent"*] and then analyze this form by using the mechanism for Wff modifiers described earlier. As noted earlier, however, this cannot handle an NP like *open door policy* where the adjective modifies the noun modifier rather than the head noun. This approach also cannot properly handle so-called "non-inherent" or "non-intersective" adjectives (a *drunken rage* is not a "rage that is drunken," i.e., the intersection of the set of rages and the set of drunken things). In a sublanguage, however, such adjectives could be assigned to special-purpose predicates having the proper selections and then decomposed later into their underlying relationships ("a rage of a drunken person").

Since the *RT* grammar uses the same embryonic Wff form for both unary adjectives (*open door*, *burnt match*) and passive participial verbs (*opened door*, *burned hand*), care must be taken in the lexicon that adjectives and verbs having the same root form receive different PROTEUS regularizations. TINSEL first attempts to interpret the modifier as a unary Wff (*door that is open*, *match that is burnt*), and if unsuccessful then attempts a binary interpretation (*door that something opened*, *hand that something burned*).

TINSEL only attempts a unary analysis of progressive verbal modifiers (*flashing light* = "light that is flashing"). There are rare exceptions in which the verb is transitive, as in the RAINFORM *relieving aircraft* ("#aircraft that is relieving"), but if TINSEL relaxed its rule it would also have to accept a wide array of ill-formed phrases like *#containing box* or *#including book*, so at present it cannot handle the exceptions. The interpreter again assumes that progressive statal adjectives (*amusing person*) have been given PROTEUS regularizations that cannot be confused with progressive actional verbs (*laughing person*), since the verb *amuse* calls for an obligatory object but the adjective *amusing* does not. It is also assumed that compound participial adjectives (*hand-*

tooled glove, man-eating tiger) have been regularized as single Preds.

Since the *RT* grammar does not attempt to distinguish progressive verbal modifiers (*parking car*) from noun modifiers (*parking difficulty*), TINSEL tries both possibilities, analyzing the first as *car that be parking* and the second as a nominalization of *parking is difficult*.

Other N-N Modifiers

Predicate-argument type checking is not the proper arena for analyzing N-N relations on head nouns not normally modeled as argument-taking predicates, because considerable world knowledge is usually needed to understand such constructs. Still, TINSEL allows nouns to be given *ad hoc* slots for that purpose if the sublanguage is sufficiently constrained for it to be practical. Examples of such N-N relations from the RAINFORM data are

TASS units	[units equipped with TASS]
P3 aircraft	[aircraft subclass of P3]
buoy barrier	[barrier composed of buoys]
attack solution	[solution used in attack]
helo LOFAR	[LOFAR deployed by helicopter].

In these examples, the implicit relation slots :equipment, :subclass, :composition, :use and :deployer could be provided to the sort predicates of the appropriate head nouns, and mapped to with the AN-STG Role marker. If the noun class has no argument slots, however, TINSEL assumes that N-N interpretation is to be ignored and just passes the noun modifier along uninterpreted for later analysis by a bona fide NP-handling component, which is probably the better strategy.

Computing Denotations of NPs

Some NPs have denotations that cannot be derived directly from the head noun, as in the examples *Ross polished/#drove most of his car* and *#The melted ice shattered* discussed earlier. Another example is from McCawley [13]:

- (a) I subtracted 13.5 from 91.2.
- (b) I subtracted what I had just computed from 91.2.
- (b') #I subtracted what I had just baked from 91.2.

If (b') were to regularize as

```
(PAST SUBTRACT I
  (WHICH Y1 THING SINGULAR (PAST PERF BAKE I VAR))
  (FROM 91.2)),
```

TINSEL (using its underspecification mechanism) would find no fault with it, since a **thing** can be baked and a **thing** can also be subtracted. As an alternative, we have developed an experimental version of TINSEL that replaces the underspecific semantic class of the head noun by the semantic class of the relative clause's selectional constraint (**food**), allowing (b') to be rejected. This cannot, however, handle Jackendoff's [12] more complex example

```
I ate something that was the result of what Bill acknowledged to be a new
  baking process.
#I ate something that was the result of what Bill acknowledged to be a
  syntactic transformation.
```

where the denotations of the NPs *what Bill acknowledged to be x* and *the result of x* vary with the value of the argument *x* and can be determined only by application of real-world knowledge, not semantic well-formedness constraints.

In our prior work using the LSP system, attribute computation was sometimes used during selection to assign an NP a semantic class other than that of its head noun, depending on the semantic class of its modifiers. This was used primarily to interpret certain modified NPs as state-predicating, as in

- (1a) We experienced a problem.
- (1b) #We experienced a valve.
- (1c) We experienced a broken valve.

where we want the NP *a broken valve* to be assigned the same semantic class (e.g., **situation**) as a word like *problem* so that both will correctly pass selection as the object of *experience*. However, the NP *a broken valve* can have an **equipment** interpretation as well:

- (2a) We replaced a broken valve.
- (2b) We replaced a valve that was broken.
- (2c) #We experienced a valve that was broken.

Deciding in what circumstances an NP denotes a situation or state rather than a physical entity is a problem that properly belongs in pragmatic analysis, since it involves such complex issues as the presence of the indefinite article (*#We experienced the broken valve*) and whether the implicit state is worthy of predication (*We have water in the oil sump*, *#We have water in the ocean*). In addition, TINSEL (unlike LSP) is faced with more than just assigning semantic class to a phrase for selection purposes; it must generate an appropriate INSOOP expression representing the phrase's meaning, and it is not clear what the logical form of the object of (1c) should be. For those reasons, attribute computation has not been incorporated into the interpreter. The object of a verb like *experience* must instead be given a relaxed selectional constraint that will not only accept both (1a,c), but also (1b) as well.

The NP complements of verbs like *experience* also tend to be those that can occur as sentence fragments in Navy messages. For example, the sentence fragment (3a) can be understood as *Valve is broken* (i.e., *There is a broken valve*), whereas no implicit predicate can be recovered (in a message context) from the ill-formed sentence fragment (3b):

- (3a) Broken valve.
- (3b) #Valve.
- (4a) Water in oil sump.
- (4b) #Water.

Similarly, the NP sentence fragment (4a) is understandable whereas (4b) is not. If the grammar were to regularize (3a) in such contexts as *breakage of valve*, the phrase would receive an interpretation (as a **break** event) acceptable to the selection constraint on the object of *experience*. By the same token, regularizing (4a) as *Water is in oil sump* would allow it to be interpreted as a **location** predication rather than an instance of **water**. It is not clear that this approach is linguistically justified, however. We have implemented an experimental version of TINSEL that generates the alternative interpretation of such forms without help from the grammar or regularization component, but only pragmatics can resolve whether the resulting interpretations are really "predication-worthy."

ARGUMENT LABELING

Although originally inspired by case grammar, TINSEL does not incorporate the generalized mapping rules from syntactic role to thematic role that Fillmore proposed. To understand why, we need

to consider more closely just what argument labels like :agent, :from-poss and :to-poss really mean.

The Case-Slot Identity Theory

Eugene Charniak, in his "case-slot identity theory" [14], noted the similarity between Fillmore's case frames and the frames used as meaning representations in AI. He proposed that there might be a direct mapping between the two, giving them an equivalence (identity) relation. If both Fillmore's and Charniak's theories were correct, all that would be needed to represent the logical form of sentences in a natural language would be to determine what the cases of the language are, derive case frames for each verb, and map directly from syntax to meaning representation. For example, if the verb *buy* takes the cases :agent, :patient, :from-poss and :price (*Nadia bought the marmot from Ross for a dollar*), then in this theory a **purchase** frame — with four slots equivalent to the four cases of the verb — could be used as a meaning representation for *buy* sentences. The theory thus moves beyond regarding case frames as a bridge or intermediate representation to regarding them as a target meaning representation, and is the approach Hirst has adopted for his semantic interpreter Absity [15].

There are several problems with the case-slot identity theory, some noted by Charniak at the time and also discussed in a follow-up paper by Fillmore [10]:

- In the twenty years since Fillmore's original paper was published, there is still no universal agreement on the set of English cases.
- The **purchase** frame used for *buy* cannot simultaneously represent the verb *sell*, because although the underlying concept of a commercial transaction is the same and involves the same participants (buyer, seller, merchandise, and money), the case frames of the two verbs are different: the :agent of *sell* is the :from-poss of *buy* and the :agent of *buy* is the :to-poss of *sell*.
- Verbs often do not correspond to simple predicates but to complexes of predicates, and acquire their complement behavior from each member of the complex. For example, *Ross hurt his foot on a rock* has a :to-loc participant not because the predicate **injure** does, but because **put** does: Ross put his foot on a rock, and that action injured his foot. A simple **hurt** frame would not capture this decomposition.

Fillmore's analysis of these problems is that they all derive from the attempt to equate case role with semantic role — in other words, the case-slot identity theory. Verbs and their case frames are just "windows" onto underlying event scenarios, says Fillmore, not all-encompassing descriptions built from semantic primitives. Allen agrees, thus rejecting case-slot identity:

In many systems the cases and frame roles are collapsed to the same set and a single representation is used for both lexical knowledge and the final representation. Because of the similarities between the representations, this is a tempting thing to do. It is not clear, however, whether such a conflation of representations is wise, since quite different information is being represented [9].

Cases vs Thematic Relations

By not attempting to encode generalized case mappings from Roles to frame slots, TINSEL lets the user experiment with semantic normalizations beyond the synonym handling that case grammar supports. In particular, the user can map inverses like *buy* and *sell* to a common frame by treating slots not as case roles but as thematic relations [12], which are roles that do not have syntactic correlates as case roles do and that are thus more purely semantic in nature. Whereas a given constituent can only fill a single case, it can fill more than one thematic relation: *Nadia* fills the :agent case in *Nadia sold*

Ross the marmot, but she also fills the :from-poss thematic relation, the latter deriving not from the syntactic behavior of the verb *sell* but only from knowledge of its semantics. The normalization of *buy* and *sell* to the same frame is made possible in TINSEL by abandoning the :agent case and using the :from-poss and :to-poss relations as the predicate slot names:

```
(defframe purchase :to-poss person :from-poss person :price money
               :item physobj)
(defpred BUY (purchase :to-poss (S) :from-poss (FROM &) :price (FOR &)
                       :item (O)))
(defpred SELL (purchase :to-poss (TO &) :from-poss (S) :price (FOR &)
                      :item (O))).
```

The same approach could be used to normalize *Ross gave the scouts money* and *The scouts received money from Ross* into the same predicate, by analyzing *Ross* as the :from-poss thematic role in both (rather than the :agent case of *give*).

Beyond Synonymy and Inverses

Although the verbs *fire* and *attack* do not have the same case frames, as illustrated by the sentences

```
We fired a torpedo at the sub.
We attacked the sub with a torpedo.
#We attacked a torpedo at the sub.
#We fired the sub with a torpedo.
```

they do have the same semantic participants of attacker, target, and weapon. Thus, it is tempting to consider treating these participants as "roles" and mapping both verbs directly to a common frame. This would not be a linguistic (case) frame, but a frame more in the AI sense of the term: a collection of related information describing a particular event or state of affairs.

This is probably not a wise move, however. Distributional analysis places *fire* in the same selection set as verbs like *launch* and *discharge*, all taking **weapon** direct objects and **org** at phrases. It assigns *attack*, on the other hand, to a selection set that includes *bombard* and *strike*, taking **org** direct objects and **weapon** with phrases. The distributions are telling us that these are really two fundamentally different kinds of actions, the first expressing physically and the second interpersonally. The semantic connection between *fire* and *attack* is not one of synonymy or even selection-set equivalence, but inference: if X fires a weapon at Y, X is probably attacking Y — but not necessarily. It is generally wiser to execute such inferences explicitly at a later stage of interpretation than to assume that two predicates are semantically equivalent just because they involve what the reader perceives to be the same participants.

COMPARISON TO OTHER APPROACHES

Linguistic String Parser

Most of the differences between TINSEL and LSP have already been discussed and can be summarized as follows.

- LSP executes selection by using syntactic (verb-subject-object, preposition-complement-host, noun-adjective, and noun-noun) patterns of semantic classes, whereas our interpreter first maps from syntax to argument role before enforcing selectional constraints.

- Selection in LSP is executed directly on lexical attributes in the phrase structure tree, but TINSEL interprets syntactic/lexical translations by mapping them to predicate models.
- LSP semantic categories are nonhierarchical word classes; TINSEL frames combine sort, predicate-argument relation and domain specification into a single data structure that supports inheritance of all three aspects.
- The LSP information formatting component, which transforms the phrase structure tree into an information format tree geared toward information retrieval applications, is distinct from the selection component and can only be applied when parsing is completed; in our interpreter the selection component constructs an application-neutral semantic representation during parsing.

NP Conjoinings

The LSP parser applies selection to syntactic (subject-verb-object) roles rather than thematic roles, so it cannot rule out NP conjoinings based on selection against the host verb as TINSEL does. Instead it allows NPs to conjoin only if their semantic classes are the same or "similar," the latter defined by using an equivalence class mechanism. This approach can suffer from overgeneralization, however. For example, the CASREP sentence *Low lube oil and fail to engage alarms sounded* required that the nouns *oil* and *alarm* be placed in different semantic categories to disallow the bad conjoining "Oil and alarms sounded," even though acceptable conjoinings like *We inspected the oil and the alarm* can be imagined. TINSEL's use of explicit predicate roles reduces the need for such a mechanism: *oil* and *alarm* can be co-arguments in those environments in which their interpretations can fill the same slot of a host predicate (like *inspect*), and by the same token they cannot pass selection as co-arguments of a predicate like *sound*.

In a sentence like the RAINFORM *Crew aborted due to lack of comm and aircraft problems*, however, predicate-argument selection might not be appropriate for ruling out the bad conjoining "communications and problems," since the user might want *lack of problems* to be considered just as well-formed as *lack of comm*. Here, more detailed pragmatic knowledge (or an equivalence heuristic like LSP's) could be brought to bear in later processing to resolve the ambiguity.

Semantic Representations

The other major distinction between TINSEL and LSP lies in the difference between predicate-argument frames and the LSP information format. Information formats are also frames in the sense that they capture a pattern of related information in a slot-filler structure. Each format represents not a single predicate-argument structure, however, but the sequence of information in an entire, prototypical sublanguage sentence. The slots of the format represent sublanguage-specific "information categories" that are often in one-to-one correspondence with the sort categories used for selection, but not always:

Because of the way in which the format is constructed, there is a close correspondence between word class membership and format column. However, for the cases where there is not a one-to-one correspondence between word class and format column, syntactic information is required in order to determine in which format slot a word should be placed [6].

For example, a format from the radiology domain might represent the sentence *Chest x-ray showed metastasis in chest* with the pattern *TESTLOC TESTN BE-SHOW MED-FIND POS PT-BODY*. Here the word *chest* (in semantic class **body-part**) can be mapped to either of the column headings *TESTLOC* or *PT-BODY* depending on the syntactic environment in which it occurs. If there were only a single format slot *BODY-PART*, the format line could not correctly represent the sentence, containing as it does two items of the same semantic class.

Since the format slots cannot be identical to the semantic categories of the domain, and yet do not represent domain-independent conceptual relations (i.e., predicate-argument roles), they are a third type altogether, a role/class hybrid that is well suited for sublanguage-dependent information retrieval, but not for AI applications. Whereas the format is a tabular structure that organizes lexical data (English words and phrases) for text retrieval, TINSEL structures are application-neutral semantic interpretations in which all lexical information has been translated into conceptual sort/predicate and role tokens. By retaining an abstract, application-independent logical structure, TINSEL representations are more suitable for generalized inferencing and other knowledge-based processing.

Question-Answering System

Bundled with an earlier version of the PROTEUS software was an example application called the Question-Answering System (QAS), a demonstration natural language interface to a student-transcript database query system. QAS contained a selection component similar to TINSEL, accomplishing selection by matching PROTEUS regularizations against declarative verb and noun models. The module could only be invoked postparse, however, and so could not be used to constrain the parser's search. Since we also wanted to experiment with several features (such as type hierarchies and role-marked predicate structures) that QAS does not support, we chose to design TINSEL from scratch. The major differences between TINSEL and the QAS interpreter are as follows.

Intermediate representation. The matching of SOOP expressions to predicate models in QAS does not result in the construction of an intermediate structure. After selection has succeeded on a Wff, QAS immediately passes the Wff and matching verb model on to a quantified-expression builder for construction of a fully scoped, first-order predicate logic expression. TINSEL instead generates the lexically/thematically interpreted INSOOP expression as its output, leaving the construction of a final meaning representation to later processing.

Sort hierarchy. The simplicity of the QAS predicate domain (student transcripts) does not require the provision of sort hierarchies. Still, the effect of a type hierarchy can be achieved in QAS by assigning words to more than one semantic class, such as mapping the Npred TEACHER to both classes **teacher** and **person**.

Uniform treatment of predicates and types. Because the QAS domain does not include high-order verbs like *believe* that can take other Wffs as arguments, QAS does not need to model predicates as typed entities themselves. Similarly, the QAS interpreter never treats noun modifiers as arguments (*teacher of linguistics*), so the classes to which Npreds are mapped do not need to be frames, just sort tokens. In TINSEL, as we have seen, both relational predicates and sort predicates are typed slot-filler structures.

Attachment of constraints. The arguments of QAS predicates are distinguished by position rather than by role labeling, but otherwise the mapping of verbs to predicates (with declarative verb models) is similar to ours. In QAS, however, the selectional constraints are attached to the mapping structures, not to the predicate slots. This means that QAS can map verbs with different domain constraints but otherwise synonymous meanings, such as *Ross ate brownies* and *Snoopy fed on Alpo* (where *feed* selects only for nonhuman subjects), to the same predicate. In TINSEL, *feed* and *eat* would have to map to distinct predicates **animal-eat** and **eat**, but the first could be a subclass of the second in the :isa hierarchy, thus identifying it as a variety of **eat**.

Thematic roles. Since QAS predicate arguments are not labeled, considerable laxity in semantic normalization is possible: *Jane enrolled in linguistics* and *Jane took linguistics* can be mapped immediately to the same predicate **take**. In the restricted domain and task this is fine, although in a broader context we have seen that such normalizations can introduce inferencing errors (if Jane enrolled but then dropped out before the first class, she did not really **take** the course).

Semantic ambiguity. Because the QAS task does not require either a discourse or world-knowledge component for resolving contextual ambiguity, the QAS representation does not need to accommodate ambiguous interpretations. In real-world domains that permit ill-formed, ambiguous or elliptical input, however, further resolution by contextual knowledge often becomes necessary and the logical form must explicitly represent the candidate meanings, as does TINSEL.

NP modifiers. Finally, QAS can interpret only those NP-postmodifying PPs that can be paraphrased as implicit BE relative clauses (which it does by consulting BE verb models as TINSEL does), and it does not deal with premodifiers.

Other Approaches

Allen. Allen's logical forms are based on case grammar, which was also a driving force behind TINSEL. He models semantic interpretation as a rule-based procedure operating on parse trees, allowing the possibility of interleaved interpretation using the lambda calculus. The PROTEUS regularization component uses lambda calculus and Translation Rule Language to generate representations that closely resemble Allen's logical form in structure. Thus, our interpreter does not have to specify the structure of the logical form for the most part, but just map Preds and Npreds onto predicates and Roles onto argument labels. Finally, TINSEL represents these mappings and selectional constraints as declarative patterns rather than in a rule-based formalism.

Absity. Hirst [15] employs a frame representation similar to ours, and like us, he attaches selectional constraints to the frame slots rather than associating them with the verb. Also, like us, he does not attempt to encode generalized mappings, but uses Verb Knowledge Packets (virtually identical to defpreds) to map deliberately from individual verb operands to frame slots. His motivations for this approach are, first, that it allows normalizations of inverses like *buy/sell*, and second, his belief that *English verbs are too idiosyncratic in their case behavior for generalized mappings to be useful*. Hirst overtly embraces the case-slot identity theory, however, and we have already seen that the *buy/sell* normalization cannot be done without abandoning certain case roles, such as :agent. Hirst himself is inconsistent on this point, sometimes claiming an :agent role for the **purchase** frame and sometimes not.

TEAM. TEAM [16] is a transportable natural language database interface system that generates a quantifier-scoped logical representation similar to the final output of QAS. Predicates and argument constraints are encoded as conceptual schema consisting of sort and nonsort predicates, the former organized into a hierarchy. TEAM also includes pragmatic (but not discourse) knowledge and a schema translator for the final mapping of logical representation to database query. NP adjuncts (e.g., *the country in Europe with the highest peak*) are mapped to vague predicates handled by individual procedures, usually at the time of applying pragmatic knowledge. TEAM treats the genitive, comparative and noun-noun constructs as vague predicates as well.

DISCUSSION

We conclude with a discussion of several issues in lexical/thematic interpretation that TINSEL does not yet fully address, and which we plan to investigate for possible future incorporation into the interpreter.

Interpreting Incomplete Forms. It was seen earlier that TINSEL achieves worthwhile time savings even though it only does selection on complete Wffs and Quants, postponing interpretation of incomplete or partial forms such as infinitive and progressive verb phrases. Modifying the interpreter to do partial semantic interpretation of incomplete forms might increase its efficiency even more, particularly if the syntactic grammar being used contained a constituent for VP (verb phrase), which our grammars generally do not.

Generalized Mappings. Like Hirst's interpreter Absity, TINSEL requires that each Pred and Npred explicitly name the Roles it will accept and which arguments each Role can map to. It does not attempt to encode generalized mapping rules like those case grammar claims, primarily because the PROTEUS subcategorization component already requires that each verb specify in detail the prepositions it will accept as complement labels, so lexical entry definition and defpred definition can go hand in hand. Certain mappings, however, such as from S and BY to :agent and from O and OF to :theme or :patient, are so common that it might be desirable to make them defaults, to be explicitly overridden by the user if desired.

Preference Semantics. Selection in TINSEL is a "hard" decision procedure that simply accepts or rejects analyses. Preference semantics [17], on the other hand, prioritizes the search space instead of pruning it, allowing semantically suboptimal or "fuzzy" expressions to receive interpretations as well as preferring one well-formed interpretation over another. Such a strategy could prove essential in creating truly robust systems that can deal with ill-formed input. However, care must be taken that this does not overly compromise the selection component's ability to constrain and direct the parser.

Alternatives to Polysemy. TINSEL currently requires that polysemous words be explicitly mapped to multiple predicates. An example we have seen already is the noun *contact*, used four different ways in the RAINFORM domain. Alternatively, some systems (including TEAM) use the technique of type coercion to reduce the amount of polysemy that must be explicitly represented, and such a technique could also prove useful in TINSEL. In particular, it might be a good approach for dealing with the jargon that is so common in military messages. One common form of jargon is the unconventional use of a word as a shorthand way of referring to a related concept, such as saying *Passed the contact to relief* instead of *Passed information about the contact to relief ship*. Here we want to coerce one item of type **event** (*contact*) into type **data**, and another (*relieve*) into type **craft**. A technique like marker-passing [15] might be used to find the requisite semantic connections between the source and target concepts. Like preference semantics, however, care must be taken that a coercion facility does not overly weaken the discriminative power of the selection component.

Order Phenomena. In the sentence *Ross went to the basement for Nadia for Alice*, most readers interpret *Nadia* as the :goal and *Alice* as the :beneficiary rather than the obverse. Similarly, *Send this to Paris to Ms. Smith* is well-formed whereas *#Send this to Ms. Smith to Paris* is not. TINSEL is not sensitive to order phenomena like these, unless they derive from the syntactic distinction between verb complements and adjuncts, which TINSEL does handle.

Adverbs. Because the SOOP syntax does not include a treatment of adverbs, TINSEL passes them along uninterpreted. The interpreter could handle many adverbs if they were mapped to RoleTerms, such as treating *Lsft arrived onsta* the same as *Lsft arrived on station*, *Detected sub visually* as *Detected sub by vision*, etc.

Attachment of Selectional Constraints. Both TINSEL and Absity associate type constraints with the slots of frames, a common approach in object-oriented or frame-based AI systems. QAS, however, associates those constraints with the verbs themselves, which seems more linguistically appropriate and allows greater flexibility in mapping and normalization, although it involves more duplication in verb model coding. This is an approach that might be given further consideration in our system.

Attributive Adjectives. Adjectives representing attributes like color, race, size and nationality act like inner cases in that they can only be filled once (*#a red blue ball*), since their values are mutually exclusive scalars. Predicative adjectives, on the other hand, represent claims that can be contradictory but are not mutually exclusive (*a smooth bumpy road* is smooth as bumpy roads go). In this regard, the original SOOP definition distinguishes two kinds of Amods (adjective modifiers): Apreds

(adjectival predicates) and Role-Attribute pairs. For example, a *ripe red apple* would be regularized as (SOME Y1 APPLE SINGULAR RIPE (COLOR RED)), where RIPE is an Apred and (COLOR RED) is a Role-Attribute pair. Because the adjective's lexical entry determines which type of Amod regularization it receives, an ambiguous adjective like *green* that has both attributive (color) and predicative (unripe) interpretations must be given two lexical entries, generating two distinct parses from which the interpreter can select. TINSEL, however, currently treats all adjectives as predicative. The definition of AnStg could be extended to include Role-Attribute pairs, and TINSEL could be modified to map them to frame slots.

Conjunction Expansion. Our PROTEUS grammars distribute conjoined predicates over their arguments, for instance regularizing the semantically ill-formed *#Nadia hosted and bounced a ball* as the expanded *Nadia hosted a ball and bounced a ball*, which TINSEL can then (incorrectly) interpret as "Nadia hosted a dance party and bounced a spherical toy." Because both copies of the object have the same identifier variable, TINSEL could be modified to do sentence-local checking for operand copying of this sort and allow only a single interpretation to be constructed and hold true for both copies. Similarly, PROTEUS expands *#the thrown ball and game* into *the thrown ball and the thrown game*, which TINSEL can misinterpret as "the tossed ball and the deliberately lost game." The interpreter cannot recognize this as a distributed conjoining, however, because PROTEUS does not assign identifier variables to Wffs. Hence, TINSEL should perhaps do such argument copying rather than the parser.

REFERENCES

1. E. Marsh, J. Froscher, R. Grishman, H. Hamburger, and J. Bachenko, "Automatic Processing of Navy Message Narrative," NRL Report 8893, Aug. 1985.
2. K. Wauchope, M.K. DiBenigno, and E. Marsh, "Automated Text Highlighting of Navy Equipment Failure Messages," NRL Report 9154, Nov. 1988.
3. D. Perzanowski and B. Potter, "InterFIS: A Natural Language Interface to the Fault Isolation Shell," Naval Research Laboratory, unpublished manuscript.
4. R. Grishman, "PROTEUS Parser Reference Manual," PROTEUS Project Memorandum #4, Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, July 1986.
5. L. Bloomfield, *Language* (Holt, Rinehart and Winston, New York, 1933).
6. N. Sager, *Natural Language Information Processing: A Computer Grammar of English and its Applications* (Addison-Wesley, Reading, MA, 1981).
7. C.J. Fillmore, "The Case for Case," in *Universals in Linguistic Theory*, E. Bach and R.T. Harms, eds. (Holt Rinehart and Winston, New York, 1968), pp. 1-88.
8. J.M. Gawron, "Syntactic Regularization in PROTEUS," PROTEUS Project Memorandum #5, Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, March 1986.
9. J. Allen, *Natural Language Understanding* (Benjamin/Cummings, Menlo Park, CA, 1987).
10. C.J. Fillmore, "The Case for Case Reopened," in *Syntax and Semantics 8: Grammatical Relations*, P. Cole and J. Sadock, eds. (Academic Press, New York, 1977).

11. R. Grishman, *Computational Linguistics: An Introduction* (Cambridge University Press, Great Britain, 1986).
12. R.S. Jackendoff, *Semantic Interpretation in Generative Grammar* (MIT Press, Cambridge, MA, 1972).
13. J.D. McCawley, *The Syntactic Phenomena of English, Volume 1* (University of Chicago Press, Chicago, IL, 1988).
14. E. Charniak, "The Case-Slot Identity Theory," *Cognitive Science* 5(3), 285-292 (1981).
15. G. Hirst, *Semantic Interpretation and the Resolution of Ambiguity* (Cambridge University Press, Great Britain, 1987).
16. B.J. Grosz, D.E. Appelt, P.A. Martin, and F.C.N. Pereira, "TEAM: An Experiment in the Design of Transportable Natural-Language Interfaces," *Artificial Intelligence* 32, 173-243 (1987).
17. Y. Wilks, "An Intelligent Analyzer and Understander of English," *Comm. ACM* 18(5), 264-274 (1975).